Available online at www.sciencedirect.com
SCIENCE DIRECT
Mathematical and Computer Modelling 43 (2006) 1172–1188
MATHEMATICAL AND COMPUTER MODELLING
www.elsevier.com/locate/mcm

Issues in the real-time computation of optimal control

I. Michael Ross[a,*], Fariba Fahroo[b]

[a] *Department of Mechanical and Astronautical Engineering, Naval Postgraduate School, Code AA/Ro, Monterey, CA 93943, United States*
[b] *Department of Applied Mathematics, Naval Postgraduate School, Code MA/Ff, Monterey, CA 93943, United States*

Received 10 March 2005; accepted 4 May 2005

Abstract

Under appropriate conditions, the dynamics of a control system governed by ordinary differential equations can be formulated in several ways: differential inclusion, control parametrization, flatness parametrization, higher-order inclusions and so on. A plethora of techniques have been proposed for each of these formulations but they are typically not portable across equivalent mathematical formulations. Further complications arise as a result of configuration and control constraints such as those imposed by obstacle avoidance or control saturation. In this paper, we present a unified framework for handling the computation of optimal controls where the description of the governing equations or that of the path constraint is not a limitation. In fact, our method exploits the advantages offered by coordinate transformations and harnesses any inherent smoothness present in the optimal system trajectories. We demonstrate how our computational framework can easily and efficiently handle different cost formulations, control sets and path constraints. We illustrate our ideas by formulating a robotics problem in eight different ways, including a differentially flat formulation subject to control saturation. This example establishes the loss of convexity in the flat formulation as well as its ramifications for computation and optimality. In addition, a numerical comparison of our unified approach to a recent technique tailored for control-affine systems reveals that we get about 30% improvement in the performance index.

1. Introduction

The continued exponential increase in computational capabilities has brought into focus the possibility of rapid and real-time trajectory generation [1–6]. Real-time trajectory generation is an enabling technology for the management of complexity in autonomous systems. For example, in the control of mobile robots, rapid trajectory generation can be used in a feedforward mode to support the quick planning and re-planning of motion. In the case of multi-agent systems, real-time trajectory generation is a necessity for autonomous operations, to prevent collisions and/or maintain formations. Arguably, the prime driver for real-time trajectory generation is the military as it sees these technologies as enablers for force multiplication: that is, a mechanism or process by which a military force, with a fixed human capital, can achieve greater strength. As an example, situational awareness can be obtained by fusing data from a swarm of micro-aerial vehicles.

Thus, future warfare is more likely to be driven by a coordinated command and control of multi-agent heterogeneous systems consisting of a system of autonomous underwater vehicles, unmanned ground vehicles, unmanned air

* Corresponding author.
E-mail addresses: imross@nps.edu (I.M. Ross), ffahroo@nps.edu (F. Fahroo).

0895-7177/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mcm.2005.05.021

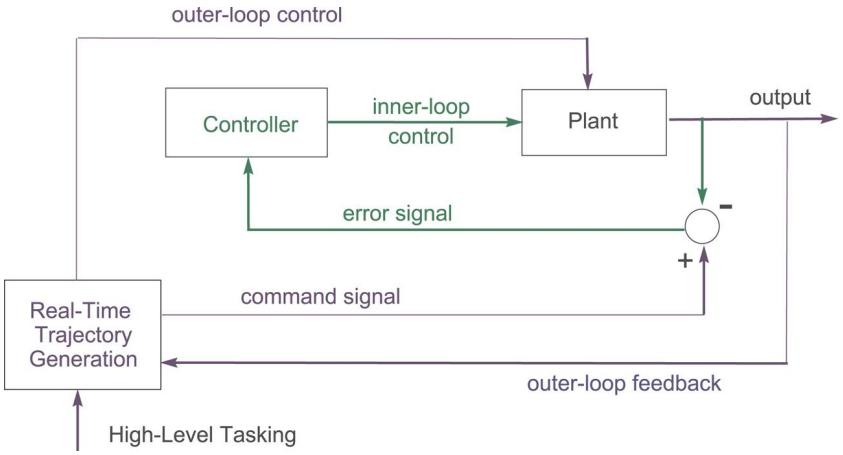

Fig. 1. A typical control system architecture for autonomous systems.

vehicles, orbiting spacecraft and other agents. Common tasks to be performed are motion planning, obstacle avoidance, target tracking, evasive maneuvers, search operations and other high-level decisions. A typical control system architecture envisioned to conduct such operations is shown in Fig. 1. Under this concept, traditional linear or nonlinear control theory would be used to design the inner loop while the outer loop would be used for rapid or real-time trajectory generation. In a purely feedforward mode, system trajectory generation would simply be used to generate the commands to alleviate the burden on the inner-loop requirements and/or to enhance the performance of the control system by providing the outer-loop control as well. A feedback of the outer loop supports a more efficient management of complexity, but requires real-time trajectory generation as indicated earlier.

In addition to meeting the demands of rapid re-planning of missions, not only must the outer loop be executed in near real time, it must also support high performance demands imposed by stringent system requirements. While optimal control theory provides a framework for tackling many of these complex problems, it is widely recognized [1,3,4,7–10] that solving these problems is extremely difficult. Much of the difficulty arises in the search for closed-form solutions to optimal feedback laws. Despite advancements in computational power and viscosity methods, the Hamilton–Jacobi framework continues to be beset with fundamental problems such as the "curse of dimensionality" and the nonsmoothness of the Bellman function [11,12]. To circumvent these difficulties, a number of approaches have been proposed, particularly in recent years, for solving the problem of real-time trajectory generation. Some recent techniques extend and explore the notion of way-points and the concept of operations envisioned for a given autonomous vehicle [1,2]. Perhaps one of the most sophisticated ideas advanced for real-time trajectory generation is the exploitation of the concept of differential flatness, proposed by Fliess and his colleagues [7,8,13,14]. It has been argued in [3,4,9,15] that computational schemes based on utilizing the property of flatness of a system are fundamentally superior to well-known collocation methods [16,17] applied to the same system. This argument is quite sound since smooth curves in flat space are automatically feasible with respect to the dynamics; therefore in a differentially flat formulation the number of variables and constraints reduces considerably and this in turn can potentially lead to considerable reduction in computational time. However, recent research [18] reveals that the ease in computation offered by the reduction of the number of constraints in flatness-based methods can also be achieved by exploiting sparse linear algebra in solving the underlying optimization problem. It turns out that real-time trajectory generation is not merely driven by the number of constraints or optimization variables; rather, the problem is fundamentally intertwined with optimization principles, such as convexity, as well as control-theoretic concepts such as flatness. Regardless of these issues, many of the proposed tools and techniques are typically not portable across equivalent mathematical formulations of the same dynamical system or, alternatively, to heterogeneous multi-agent systems that do not share differential-geometric topologies or concept of operations.

Given that real-time computation of optimal controls is an enabling technology for solving a vast array of complex control problems, it is essential that the cause and effect of the competing principles be isolated [19]. One clear approach to this analysis is to design a unified framework for solving nonlinear optimal control problems that includes the range of possibilities from differential flatness to differential inclusions. In this paper, we present such a framework.

We first articulated our unified framework in [20]; here, we extend and clarify the concepts introduced in that paper.

While a unified framework is not only necessary to isolate and systematically analyze the issues raised above, it is worth noting the dollar-cost savings achievable from such an implementation. For example, in the case of multi-agent systems, if agents of a heterogeneous system have a common software, complex decisions can be quickly reprogrammed across the board. The alternative of specialized software (even when common over homogeneous agents) not only increases the cost of implementation, it also increases the risk of failure as a specialized software may not be able to handle unforeseen scenarios, particularly as a result of heterogeneous interactions. In addition, a common software reduces the enormous cost associated with software verification and software management. Simply put, a unified framework is strongly motivated by both theoretical and practical considerations.

As a demonstration of the major ideas promulgated in this paper, we solve a robotics problem from Isidori in [21] that was further analyzed in [15] and [22] with respect to computation. This illustrative example allows us to isolate the computational method from the various means of representing the same dynamical system so that reasonable conclusions can be drawn and numerically verified. In addition to the benefits of a unified framework already outlined, we show that we get an almost 30% improvement in the performance index when compared to the results reported in [15].

2. Fundamental ideas

A general optimal control problem can be formulated as follows [10]: Determine the absolutely continuous function, $\mathbb{R} \supset [\tau_0, \tau_f] \ni \tau \mapsto \boldsymbol{x} \in \mathbb{R}^{N_x}$, and, possibly, the clock times, τ_0 and τ_f, that minimize the end-point (Mayer) cost functional,

$$J[\boldsymbol{x}(\cdot), \tau_0, \tau_f] \coloneqq E(\boldsymbol{x}_0, \boldsymbol{x}_f, \tau_0, \tau_f) \tag{1}$$

subject to the dynamic constraints,

$$\dot{\boldsymbol{x}}(\tau) \in \mathbb{F}(\boldsymbol{x}(\tau), \tau) \quad a.e.\ \tau \in [\tau_0, \tau_f] \tag{2}$$

and end-point constraints,

$$(\boldsymbol{x}_0, \boldsymbol{x}_f, \tau_0, \tau_f) \in \mathbb{E} \tag{3}$$

where, $\boldsymbol{x}_0 \coloneqq \boldsymbol{x}(\tau_0)$, $\boldsymbol{x}_f \coloneqq \boldsymbol{x}(\tau_f)$, $\mathbb{E} \subset \mathbb{R}^{N_x} \times \mathbb{R}^{N_x} \times \mathbb{R} \times \mathbb{R}$ is a given set, $E : \mathbb{E} \to \mathbb{R}$ is a given function, and $\mathbb{F} : \mathbb{R}^{N_x} \times \mathbb{R} \twoheadrightarrow \mathbb{R}^{N_x}$ is a given multifunction. In the development of necessary and sufficient conditions for optimal control problems, it is well known that one has to carefully stipulate the precise definitions of the spaces and sets involved, in addition to stating the metric for the choice of a putative minimizer. It is also well known [10,12] that, even if the data for the problem is well behaved (e.g. C^∞ functions), a solution may not exist, particularly in a preferred function space of solutions. From various existence theorems [10], the function space where existence can be assured by imposing certain conditions on the problem data is $W^{1,1}$, the space of absolutely continuous functions. In general, the space of interest in optimal control theory is a Sobolev space [23] denoted as $W^{m_x,p}([\tau_0, \tau_f], \mathbb{R}^{N_x})$, which consists of all vector-valued functions, $[\tau_0, \tau_f] \mapsto \boldsymbol{x} \in \mathbb{R}^{N_x}$, whose jth derivative is in $L^p([\tau_0, \tau_f], \mathbb{R}^{N_x})$ for all $0 \leq j \leq m_x$.

Within a computational framework, we need to find solutions that are numerically bounded. Hence, we seek solutions in a function space smaller than $W^{1,1}$. In avoiding the details of existence issues for the purposes of applications, we simply assume that a solution, $\boldsymbol{x}(\cdot)$, exists in *at least* $W^{1,\infty}([\tau_0, \tau_f], \mathbb{R}^{N_x})$, the space of Lipschitz-continuous functions. Alternatively, we assume that the functional $J : W^{m_x,\infty} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is continuous and bounded from below for some $m_x \geq 1$. It will be presented later that despite these apparently stringent assumptions, many of the tools and techniques that are widely promulgated in control theory require stronger smoothness assumptions (i.e. $m_x > 1$). In the subsequent sections, we will show that our computational method naturally exploits the smoothness inherent to the optimal system trajectories.

Frequently, we are concerned with dynamical systems parameterized by a control variable $\boldsymbol{u} \in \mathbb{U}(\boldsymbol{x}, \tau) \subseteq \mathbb{R}^{N_u}$,

$$\dot{\boldsymbol{x}} = \mathbf{f}(\boldsymbol{x}, \boldsymbol{u}, \tau) \tag{4}$$

where $\mathbb{U}(\boldsymbol{x}, \tau) \subseteq \mathbb{R}^{N_u}$ is some *state-dependent* control set, and $\mathbf{f} : \mathbb{R}^{N_x} \times \mathbb{R}^{N_u} \times \mathbb{R} \to \mathbb{R}^{N_x}$ is a given vector field. In a theoretical framework, state-dependent control spaces and other complexities can be dealt with in a unified manner by resorting to a hodograph transformation, in which case the multifunction $\mathbb{F}$ is given explicitly by

$$\mathbb{F}(\boldsymbol{x}, \tau) = \{\mathbf{v} : \mathbf{v} = \mathbf{f}(\boldsymbol{x}, \boldsymbol{u}, \tau),\ \boldsymbol{u} \in \mathbb{U}(\boldsymbol{x}, \tau)\}. \tag{5}$$

In fact, one of the arguments put forward [10,12] for considering differential inclusions as a fundamental object is precisely that it allows a treatment of such complexities. From a numerical perspective, the differential inclusion framework offers a potential decrease in computational time since there are N_u fewer variables to compute. The practical problem is that it may not be possible to analytically perform the explicit elimination of the control variables as required by Eq. (5). Thus, the ideas remain restricted to a small class of nonlinear systems such as those that are control-affine. Even when it is possible to explicitly eliminate control variables, Sussmann [24] has pointed out that a passage from an equation to an inclusion may be accompanied by a loss in information, in which case the two systems Eqs. (2) and (4) are not mathematically equivalent.

A more general framework advanced by Lowen and Rockafellar [25] that subsumes system representations given by Eqs. (2) and (4) is a *controlled differential inclusion*,

$$\dot{\boldsymbol{x}} \in \mathbb{F}(\boldsymbol{x}, \boldsymbol{u}, \tau). \tag{6}$$

With this concept, Eq. (4) may be included in Eq. (6) as the singleton $\{\mathbf{f}(\boldsymbol{x}, \boldsymbol{u}, \tau)\} = \mathbb{F}(\boldsymbol{x}, \boldsymbol{u}, \tau)$ rather than a set as in Eq. (5). The primary computational advantage of a controlled differential inclusion formalism is that we can maintain the full information represented in a vector field formulation (thus addressing Sussmann's objections) by eliminating only those control variables (see [26]) that maintain equivalence between Eqs. (4) and (6). In addition, we can further restrict the elimination of control variables to only those variables that permit an explicit hodograph transformation so that it is still possible to gain a numerical advantage. See [26] for an illustrative example. Regardless of this, it is worth noting that many new formulations of dynamics [27] are directly given in terms of a differential inclusion rather than a re-formulation of the system representation via a hodograph transformation. In other words, it is important for a unified framework to address head-on, in a computationally equivalent manner, a system representation that is given in terms of a differential inclusion, or a controlled differential inclusion, or a vector field formulation.

Motivated by computational considerations, we further explore system equivalence. For mechanical systems, a natural method for reducing variables is *not* transforming the dynamics to a state-space form. That is, since a mechanical system can be described by a Lagrangian system,

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L(\mathbf{q}, \dot{\mathbf{q}})}{\partial \dot{\mathbf{q}}} - \frac{\partial L(\mathbf{q}, \dot{\mathbf{q}})}{\partial \mathbf{q}} = \boldsymbol{u} \tag{7}$$

where $\mathbf{q}$ is the generalized coordinate, and $L : \mathbb{R}^{N_q} \times \mathbb{R}^{N_q} \to \mathbb{R}$ is the Lagrangian of the system, the essential idea is not to transform Eq. (7) to a state-space form, as this results in additional (velocity) variables as well as additional (kinematic) constraints of the form $\dot{\mathbf{q}} = \mathbf{v}_q$. Further reduction in the number of variables for a Lagrangian system may be possible by way of a *second-order differential inclusion*:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L(\mathbf{q}, \dot{\mathbf{q}})}{\partial \dot{\mathbf{q}}} - \frac{\partial L(\mathbf{q}, \dot{\mathbf{q}})}{\partial \mathbf{q}} \in \widehat{\mathbb{U}}(\mathbf{q}, \dot{\mathbf{q}}, \tau) \tag{8}$$

where the control space, $\widehat{\mathbb{U}}(\mathbf{q}, \dot{\mathbf{q}}, \tau)$, is, as indicated, allowed to be dependent on the configuration, in much the same way as we allowed state-dependent control sets in Eq. (5). Note also that nonholonomic constraints are also included in the Lagrangian inclusion. In any case, the Lagrangian inclusion comes with all the trappings and advantages of a differential inclusion as already indicated. A recent approach to exploiting the properties of Lagrangian systems is the method of controlled Lagrangians [28–30], but note that these methods are largely directed at stabilizing mechanical systems (the inner loop in Fig. 1), and not at computing optimal control (i.e. the outer loop), although some initial ideas have appeared recently in [30]. Thus, our goals and methods for the control of Lagrangian systems are separate and distinct from those of Bloch et al. [28–30].

Whether or not a control system is Lagrangian, a useful control-theoretic concept that can be exploited for the purpose of computing optimal controls is to change coordinates, $\Phi : \boldsymbol{x} \mapsto \mathbf{w} \in \mathbb{R}^{N_x}$, to a normal form, [21]

$$\begin{pmatrix} \dot{w}_1 \\ \vdots \\ \dot{w}_{r-1} \end{pmatrix} = \begin{pmatrix} w_2 \\ \vdots \\ w_r \end{pmatrix} \tag{9}$$

$$\begin{pmatrix} \dot{w}_r \\ \vdots \\ \dot{w}_{N_x} \end{pmatrix} = \widetilde{\mathbf{f}}(\mathbf{w}, u) \tag{10}$$

where r is the relative degree, and $\widetilde{\mathbf{f}} : \mathbb{R}^{N_x} \times \mathbb{R} \to \mathbb{R}^{N_x - r + 1}$ is a projection of the transformed function, $\mathbf{f}$. Deferring, for the moment, a discussion of the more general case when the number of control variables is greater than one – as implicitly assumed in the above representation – we note that a change in coordinates from $\boldsymbol{x}$ to $\mathbf{w}$ is possible when the function $\mathbf{f}$ is affine in the control variable. For non-control-affine systems, a dynamic extension may be performed to make available the tools of coordinate changes. In this case, we introduce additional dynamic constraints (integrators). The triangular structure of the differential equations in the normal form is fully triangular when $r = N_x$,

$$\begin{aligned} \dot{w}_1 &= w_2 \\ &\vdots \\ \dot{w}_{N_x} &= \widetilde{f}(\mathbf{w}, u). \end{aligned}$$

Such a structure offers a computational advantage when a numerical procedure is based on an LU decomposition.

The normal form of the equations can also be used to reduce the number of differential constraints by $r - 1$, by expressing Eqs. (9)–(10) in a higher-order form,

$$\begin{pmatrix} w_1^{(r)} \\ \dot{w}_{r+1} \\ \vdots \\ \dot{w}_{N_x} \end{pmatrix} = \widetilde{\mathbf{f}}(\mathbf{w}, u). \tag{11}$$

If $r = N_x$, the differential constraints can be scalarized to

$$w_1^{(N_x)} = \widetilde{f}(\mathbf{w}, u). \tag{12}$$

When the number of control variables, N_u, is more than one, the notion of vector relative degree [21] can be used to express the dynamics in a block triangular form which can still be exploited by LU decomposition techniques. In the same spirit, it is still possible to reduce the number of dynamic constraints via

$$\sum_{i=1}^{N_u} r_i - N_u \tag{13}$$

where $\{r_1, \ldots, r_{N_u}\}$ is the vector relative degree. The details of this process are quite similar to those in the preceding discussion of the scalar relative degree. It is quite important to note that a reduction in the number of differential constraints could potentially lead to a decrease in computation time *only if higher-order derivatives can be explicitly accounted for in a computational framework*. We will explore this point further in the next section.

As in the Lagrangian system, a reduction in the number of variables (in addition to the reduction in the number of constraints) is possible by a passage to higher-order differential inclusions. For example, the control variable in Eq. (12) can be eliminated by the set-valued representation

$$w_1^{(N_x)} \in \widetilde{f}(\mathbf{w}, \mathbb{U}(\Phi^{-1}(\mathbf{w}), \tau)) \tag{14}$$

where $\widetilde{f}(\mathbf{w}, \mathbb{U}(\Phi^{-1}(\mathbf{w}), \tau))$ is a set-valued map obtained through a transformation akin to Eq. (5) and Φ^{-1} is the map, $\mathbf{w} \mapsto \boldsymbol{x}$, which exists by assumption of a diffeomorphism on $\mathbb{R}^{N_x}$. In the same spirit, when $N_u > 1$, we can obtain

higher-order vector differential inclusions that offer a reduction in both the number of variables and that of constraints. A logical progression of this argument is the question of whether one can change coordinates in such a way that the dynamical constraints can be completely eliminated. An affirmative answer to this question was provided by Fliess et al. [7,8,13,14]. The caveat is that we must now extend our notion of a coordinate transformation as diffeomorphisms over sufficiently many integrators [14]. We briefly describe this idea but refer the reader to the two seminal papers, [7] and [14].

Let $\mathbb{R}^N_m = \mathbb{R}^N \times \mathbb{R}^N \times \cdots$ denote the product of $m+1$ (finite) copies of $\mathbb{R}^N$. A controlled dynamical system, $\dot{\boldsymbol{x}} = \mathbf{f}(\boldsymbol{x}, \boldsymbol{u})$, is differentially flat [7,14] if there exists a variable $\mathbf{y} \in \mathbb{R}^{N_u}$, called the flat output, and a function $\mathbf{c} : \mathbb{R}^{N_x} \times \mathbb{R}^{N_u}_\alpha \to \mathbb{R}^{N_u}$,

$$\mathbf{y} = \mathbf{c}(\boldsymbol{x}, \boldsymbol{u}, \dot{\boldsymbol{u}}, \ldots, \boldsymbol{u}^{(\alpha)}) \tag{15}$$

such that

$$\boldsymbol{x} = \mathbf{a}(\mathbf{y}, \dot{\mathbf{y}}, \ldots, \mathbf{y}^{(\beta)}) \tag{16}$$

$$\boldsymbol{u} = \mathbf{b}(\mathbf{y}, \dot{\mathbf{y}}, \ldots, \mathbf{y}^{(\beta+1)}) \tag{17}$$

where α and β are finite positive integers that denote the number of derivatives of the respective variables, and $\mathbf{a}$ and $\mathbf{b}$ are functions defined similarly to $\mathbf{c}$; that is, $\mathbf{a} : \mathbb{R}^{N_u}_\beta \to \mathbb{R}^{N_x}$ and $\mathbf{b} : \mathbb{R}^{N_u}_{\beta+1} \to \mathbb{R}^{N_u}$. Evidently, differentiability, particularly of the control variable, is presumed. Thus, we require the control function, $\boldsymbol{u}(\cdot)$, to be selected from the space $C^\alpha([\tau_0, \tau_f], \mathbb{R}^{N_u})$.

With the complete elimination of the dynamic constraints, flatness parametrization of a control system offers a dynamic optimization problem with fewer constraints than the differential inclusion framework. However, control constraints (particularly state-dependent control constraints) transform the control region, $\mathbb{U}(\boldsymbol{x}, \tau)$, to constraints on the flat output,

$$\mathbf{z} \in \widetilde{\mathbb{U}}(\tau) \tag{18}$$

where $\mathbf{z} = [\mathbf{y}, \dot{\mathbf{y}}, \ldots, \mathbf{y}^{(s)}]^{\mathrm{T}}$, $s = \beta + 1$ and $\widetilde{\mathbb{U}}$ denotes the transformed control space,

$$\widetilde{\mathbb{U}}(\tau) = \{\mathbf{z} \in \mathbb{R}^{N_u}_s : \mathbf{b}(\mathbf{z}) = \boldsymbol{u}, \boldsymbol{u} \in \mathbb{U}(\mathbf{a}(\mathbf{y}, \dot{\mathbf{y}}, \ldots, \mathbf{y}^{(\beta)}), \tau)\}.$$

Thus, flatness parametrization implies a path constraint on the flat output resulting from a potentially complex transformation of the original control region, $\mathbb{U}(\boldsymbol{x}, \tau)$. Since this transformation need not preserve convexity, the reduction in the number of constraints, obtained by a total elimination of the dynamics, may not outweigh the potential loss of convexity of the constraint region, resulting from a transformation of the control space to the flat output space. On the other hand, if a nonconvex (or convex) control space is transformed to a convex constraint on the flat output space, there is no doubt that flatness parametrization may offer a significant computational advantage.

Recognizing the important role of convexity in computation [31,32], we note that the issue of the preservation of convexity, or the lack of it, is not limited to flatness parametrization and the control space: it also applies to the change in coordinates associated with the normal form, the transformation of the end-point set, $\mathbb{E}$, and the transformation of the cost function. In fact, one can even make the argument that the more critical requirements on convexity are on the end-point set and the cost function, since control constraints are absent for some problems. These points are best illustrated in the example problem that follows the next section.

A final point worth noting at this juncture is the computational issue of transforming a Lagrange or a generalized Bolza cost functional [25],

$$J[\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot), \tau_0, \tau_f] = E(\boldsymbol{x}_0, \boldsymbol{x}_f, \tau_0, \tau_f) + \int_{\tau_0}^{\tau_f} F(\boldsymbol{x}(\tau), \dot{\boldsymbol{x}}(\tau), \boldsymbol{u}(\tau), \tau)\, \mathrm{d}\tau \tag{19}$$

where $F : \mathbb{R}^{N_x} \times \mathbb{R}^{N_x} \times \mathbb{R}^{N_u} \times \mathbb{R} \to \mathbb{R}$ is the running cost function, measurable with respect to τ, and $J : W^{m_x,\infty}([\tau_0, \tau_f], \mathbb{R}^{N_x}) \times W^{m_u,\infty}([\tau_0, \tau_f], \mathbb{R}^{N_u}) \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, is the Bolza cost functional, with $m_u \geq 0$. Note the functional dependence of the running cost on the velocity variable. In higher-order formulations, the running cost will be functionally dependent on higher-order derivatives as well. A purportedly unified computational framework must be able to directly deal with a Bolza cost functional, rather than transform it to a Mayer form, by the well-known

process of introducing a new state variable x_{N_x+1} and the dynamic constraint,

$$\dot{x}_{N_x+1}(t) = F(\boldsymbol{x}(\tau), \dot{\boldsymbol{x}}(\tau), \boldsymbol{u}(\tau), \tau). \tag{20}$$

Although augmenting Eq. (20) with Eq. (4) to generate new state dynamics may be mathematically equivalent, there are significant computational differences [16,17]. First, Eq. (20), introduces a new variable (and, thus, a potential increase in computational time). Secondly, the new variable must satisfy an equality constraint over the entire interval $[\tau_0, \tau_f]$, a more stringent computational requirement than the approximation of an integral. Thirdly, if the function F is nonlinear, Eq. (20) destroys convexity of the constraint set; on the other hand, if F is convex in its variables, the Bolza form preserves convexity because a convex optimization problem is the minimization of a convex cost over a convex set [31]. Finally, integration is a smoothing process (preferred computationally) while differentiation is an anti-smoothing process [33]. Thus, Eq. (19) is numerically preferable to Eq. (20). For the purposes of brevity, we will not discuss isoperimetric constraints, but it is clear that these problems get exacerbated in such situations.

What is apparent from the discussions in this section is that two mathematically equivalent representations of the same problem may exhibit fundamentally different computational properties as a result of the various transformations of the functions and the sets involved; however, a good unified computational framework must prevent exacerbating the numerical issues, and avoid biasing one transformation over another. As noted in the previous section, designing different computational methods for different dynamical systems or alternative coordinatization of the same dynamical system is an expensive proposition that we seek to avoid. In fact, a unified computational procedure must take into account any efficiencies offered by control-theoretic concepts. We will now show how the pseudospectral framework described in the next section can accommodate all these notions in an efficient manner.

3. Unified computational framework

Our unified framework is based on pseudospectral (PS) methods [34,35]. Our ideas apply across all PS methods, but we limit our discussion to Legendre PS methods for clarity. From a differential-geometric framework, PS methods are quite different from standard numerical methods (like Runge–Kutta) because they accurately separate the discretization of the tangent bundle from that of the vector field. Hence, they have wide applicability as is evident from recent applications to systems governed by differential inclusions [36], differentially flat systems [18], higher-order systems [37] and standard state-space systems [38,39]. In addition, PS methods generally treat the approximation of integrals by Gaussian quadratures which facilitates an efficient computational framework.

In the Legendre PS method, the states and controls are approximated by Nth-order Lagrange polynomials which interpolate the functions at optimally chosen nodes: Legendre–Gauss–Lobatto (LGL) points. These points, which are the extrema of the Nth-order Legendre polynomials, are optimal in the sense that they yield the least interpolation error in the L^2 sense. The other widely used [34,35,40] quadrature nodes are the Chebyshev nodes (extrema of the Chebyshev polynomials) which give the optimal interpolation error in the L^∞ norm. But the LGL nodes are more versatile since they can also be used in the discretization of the integral portion of the problem. More precisely, the LGL nodes $t_l, l = 0, \ldots, N$, which are distributed on the interval $[-1, 1]$ are defined as [34]

$$t_0 = -1, \quad t_N = 1$$

and for $1 \leq l \leq N-1$, t_l are the zeros of $\dot{L}_N$, the derivative of the Legendre polynomial, L_N. The discretization process begins by approximating the continuous state and control variables [38,42]:

$$\boldsymbol{x}(t) \approx \boldsymbol{x}^N(t) = \sum_{l=0}^{N} \boldsymbol{x}_l \phi_l(t) \tag{21}$$

$$\boldsymbol{u}(t) \approx \boldsymbol{u}^N(t) = \sum_{l=0}^{N} \boldsymbol{u}_l \phi_l(t) \tag{22}$$

where, for $l = 0, 1, \ldots, N$,

$$\phi_l(t) = \frac{1}{N(N+1)L_N(t_l)} \frac{(t^2-1)\dot{L}_N(t)}{t - t_l} = \begin{cases} 1 & \text{if } l = k \\ 0 & \text{if } l \neq k \end{cases} \tag{23}$$

are the Lagrange interpolating polynomials of order N, and $\boldsymbol{x}_l = \boldsymbol{x}^N(t_l), \boldsymbol{u}_l = \boldsymbol{u}^N(t_l)$. The derivative terms are approximated from Eq. (21) by differentiating the approximation

$$\dot{\boldsymbol{x}}(t) \approx \dot{\boldsymbol{x}}^N(t) = \sum_{l=0}^{N} \boldsymbol{x}_l \dot{\phi}_l(t) \tag{24}$$

and then evaluating the expression at the LGL nodes. This process gives rise to the $(N+1) \times (N+1)$ differentiation matrix $\mathbf{D}_1$ with entries $D_{1,kl} = \dot{\phi}_l(t_k)$ or

$$\mathbf{D}_1 := [D_{1,kl}] := \begin{cases} \dfrac{L_N(t_k)}{L_N(t_l)} \cdot \dfrac{1}{t_k - t_l} & k \neq l \\ -\dfrac{N(N+1)}{4} & k = l = 0 \\ \dfrac{N(N+1)}{4} & k = l = N \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

which operates over each component of the discretization, $\mathbf{X} = (\mathbf{x}_0; \mathbf{x}_1; \ldots; \mathbf{x}_N)$, to generate a discrete derivative $\dot{\mathbf{X}} = D_1 * \boldsymbol{X} = (\dot{\mathbf{x}}_0; \dot{\mathbf{x}}_1; \ldots; \dot{\mathbf{x}}_N)$. In general, commuting the operations of differentiation and approximation is not desirable due to the large errors resulting from the Runge phenomenon [35]; however, PS methods offer an accurate way to commute these operations by way of choosing nodes based on minimal error norms. In fact, it can be shown that PS methods offer exponential convergence (with respect to the number of nodes) to the derivative of an analytic function [35]. Letting $\mathbf{D}_1 \equiv \mathbf{D}$, the higher-order derivatives can be obtained from matrix powers: $\mathbf{D}_i = \mathbf{D}^i$. Thus, $\mathbf{D}_2$ is obtained by simply squaring $\mathbf{D}_1$, while $\mathbf{D}_3 = \mathbf{D}^3$ and so on. For a more accurate and numerically stable calculations of the higher-order derivative matrices, see [41]. Discretization of the different dynamical constraints is based on imposing the constraints only at the LGL nodes. In this manner, the functions are replaced by a vector of their values at the points and derivative operators are replaced by differentiation matrices. Therefore, discretization of the differential inclusion can be written as

$$\dot{\boldsymbol{x}}(\tau_i) \simeq \dot{\boldsymbol{x}}^N(\tau_i) = \sum_{j=0}^{N} D_{1,ij} \boldsymbol{x}_j \in \frac{\tau_f - \tau_0}{2} \mathbb{F}(\boldsymbol{x}_i, \tau_i) \qquad i = 0, 1, \ldots, N \tag{26}$$

where the factor $(\tau_f - \tau_0)/2$ comes from an affine transformation of the time domain from $[\tau_0, \tau_f]$ to $[-1, 1]$, and τ_i are the shifted LGL nodes. Similarly, the state-space constraint

$$\dot{\boldsymbol{x}}(\tau) = \mathbf{f}(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \tau)$$

is approximated by

$$\dot{\boldsymbol{x}}^N(t_k) = \sum_{l=0}^{N} \boldsymbol{x}_l \dot{\phi}_l(t_k) = \sum_{l=0}^{N} D_{kl} \boldsymbol{x}_l = \frac{\tau_f - \tau_0}{2} \mathbf{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, \tau_k) \tag{27}$$

while a general higher-order differential constraint of the form

$$\mathbf{f}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \ldots, \boldsymbol{x}^{(n)}) = \boldsymbol{u} \tag{28}$$

can be discretized as

$$\mathbf{f}(\boldsymbol{X}, \mathbf{D}_1 * \boldsymbol{X}, \ldots, \mathbf{D}_n * \boldsymbol{X}) = \boldsymbol{U} \tag{29}$$

where $\mathbf{U} = (\mathbf{u}_0; \mathbf{u}_1; \ldots; \mathbf{u}_N)$. It is thus apparent that a flat output and its derivatives can also be easily discretized as [18],

$$\mathbf{Y}_i = [\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_N][\mathbf{D}^i]^{\mathrm{T}} \tag{30}$$

where $\mathbf{Y}_i$ is an equivalent representation of the vector-valued polynomials, $\mathbf{y}^{(i)}$, $i = 1 \ldots, s$, which are given by

$$\dot{\mathbf{y}}(t) = \sum_{l=0}^{N} \mathbf{y}_l \dot{\phi}_l(t), \ \ldots, \mathbf{y}^{(s)}(t) = \sum_{l=0}^{N} \mathbf{y}_l \phi_l^{(s)}(t) \tag{31}$$

where the superscript (s) denotes the sth derivative. As noted in [18], for a flat system, this representation need not be an approximation.

For discretizing the generalized Bolza cost function, Eq. (19), the Gauss–Lobatto integration rule yields

$$J^N[\mathbf{X}, \mathbf{U}, \tau_0, \tau_f] = \frac{\tau_f - \tau_0}{2} \sum_{k=0}^{N} F\left(\mathbf{x}_k, \sum_{j=0}^{N} D_{1,kj}\boldsymbol{x}_j, \mathbf{u}_k, \tau_k\right) w_k + E(\mathbf{x}_0, \mathbf{x}_N, \tau_0, \tau_f) \tag{32}$$

where w_k are the LGL weights given by

$$w_k := \frac{2}{N(N+1)} \frac{1}{[L_N(t_k)]^2} \quad k = 0, 1, \ldots, N.$$

This formulation can easily be adapted to the cases where the cost functional is a function of higher-order derivatives of the states or outputs as well (as in the case of a differentially flat system). In these cases, the derivatives of the functions are replaced by discrete approximations resulting from a repetitive use of the differentiation matrix to generate derivatives of higher order. The end result is that the optimal control problem for all of the different formulations discussed in the previous section can be easily and accurately approximated to a nonlinear programming (NLP) problem. One important aspect of the NLP obtained by PS methods is that it preserves the structure of the original optimal control problem which is of significant consequence for the dualization of the problem [42] and convergence of the discretization [43,44]. As a matter of fact, the following theorem has been proved in [44]:

Theorem 1.1 (*Convergence*). *Let* $\boldsymbol{x}^*(\cdot) \in W^{m_x,\infty}([\tau_0, \tau_f], \mathbb{R}^{N_x})$ *be the optimal state (trajectory) associated with the optimal control,* $\boldsymbol{u}^*(\cdot) \in W^{m_u,\infty}([\tau_0, \tau_f], \mathbb{R}^{N_u})$. *Under proper technical conditions, the following convergence result holds:*

$$\|\boldsymbol{x}^*(\cdot) - \boldsymbol{x}^N(\cdot)\|_{L^\infty} = O\left(\frac{1}{N}\right)^{m_x}$$
$$\|\boldsymbol{u}^*(\cdot) - \boldsymbol{u}^N(\cdot)\|_{L^\infty} = O\left(\frac{1}{N}\right)^{m_u}.$$

From this theorem it is clear that the smoother the optimal solution, the faster the convergence of the PS solution.

Many of these ideas are further illustrated in the next section by way of a benchmark nonlinear control problem where eight different formulations are considered.

4. Example problem

In order to demonstrate our ideas in a concise manner, we choose the one-link flexible robot arm discussed in the textbook by Isidori [21]. The system has two degrees of freedom and one control input and is further examined in [15] and [22]. Although this system is static feedback linearizable and is hence trivially differentially flat, we include this formulation among the array of possibilities to demonstrate the issues regarding the transformation of the control constraint to the flat space.

4.1. Lagrangian formulation

From the Lagrangian of this system

$$L(\mathbf{q}, \dot{\mathbf{q}}) = \frac{I_1\dot{q}_1^2 + I_2\dot{q}_2^2}{2} + k_1\frac{(q_1 - q_2)^2}{2} + m_1 gl(1 - \cos q_1)$$

the Euler–Lagrange equations of motion are

$$I_1\ddot{q}_1 + m_1 gl \sin q_1 + k(q_1 - q_2) = 0 \tag{33}$$
$$I_2\ddot{q}_2 - k(q_1 - q_2) = u \tag{34}$$

where [22]

$$I_1 = I_2 = 1.0, \quad k = 1.0, \quad g = 9.8, \quad m_1 = 0.01, \quad l = 0.5.$$

The optimal control problem is minimizing

$$J_A = \int_{\tau_0}^{\tau_f} u^2(\tau)\,\mathrm{d}\tau \tag{35}$$

subject to the Euler–Lagrange equations, the end-point constraints,

$$[q_1(\tau_0), q_2(\tau_0), \dot{q}_1(\tau_0), \dot{q}_2(\tau_0)] = [0.03, 0.01, 0.04, 0.05] \tag{36}$$
$$\left[q_1(\tau_f), q_2(\tau_f), \dot{q}_1(\tau_f), \dot{q}_2(\tau_f)\right] = [0.06, 0.02, 0.08, 0.02] \tag{37}$$

and the control constraint,

$$\mathbb{U} = \{u : -15 \le u \le 15\}. \tag{38}$$

Discretization of this problem by the PS method yields an NLP variable of dimension $3(N+1)$ where N is the degree of approximation and $N+1$ is the number of nodes.

4.2. Lagrangian inclusion

In this formulation the controls are eliminated from the Euler–Lagrange formulation resulting in the following second-order differential inclusion:

$$I_1\ddot{q}_1 + m_1 gl \sin q_1 + k(q_1 - q_2) = 0 \tag{39}$$
$$|I_2\ddot{q}_2 - k(q_1 - q_2)| \le 15. \tag{40}$$

The cost function is transformed to

$$J_B = \int_{\tau_0}^{\tau_f} [I_2\ddot{q}_2(\tau) - k(q_1(\tau) - q_2(\tau))]^2\,\mathrm{d}\tau. \tag{41}$$

As a result of the linearity of the transformation, the convexity of the running cost (see Eq. (35)) is preserved [32] in Eq. (41). The end-point conditions remain unchanged from Eqs. (36) and (37). The resulting NLP variable has dimension $2(N+1)$.

4.3. State-space formulation

The generalized coordinates, q_1 and q_2, of the Lagrangian formulation are easily converted to a state-space formulation by the following linear transformation:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} q_1 \\ q_2 \\ \dot{q}_1 \\ \dot{q}_2 \end{pmatrix} \tag{42}$$

resulting in,

$$\dot{x}_1 = x_3 \tag{43}$$
$$\dot{x}_2 = x_4 \tag{44}$$

$$\dot{x}_3 = -\frac{m_1 gl \sin x_1 + k(x_1 - x_2)}{I_1} \tag{45}$$

$$\dot{x}_4 = \frac{k(x_1 - x_2) + u}{I_2}. \tag{46}$$

As a result of the linear transformation, Eq. (42), the functions representing the end-point set given by Eqs. (36) and (37) transform linearly. The related NLP variable is of dimension $5(N+1)$, and is the largest among these formulations.

4.4. Differential inclusion

In this formulation, the differential constraints are given by

$$\dot{x}_1 = x_3 \tag{47}$$

$$\dot{x}_2 = x_4 \tag{48}$$

$$\dot{x}_3 = -\frac{m_1 gl \sin(x_1) + k(x_1 - x_2)}{I_1} \tag{49}$$

$$|I_2 \dot{x}_4 - k(x_1 - x_2)| \leq 15 \tag{50}$$

while the cost function is transformed to

$$J_C = \int_{\tau_0}^{\tau_f} [I_2 \dot{x}_4(\tau) - k(x_1(\tau) - x_2(\tau))]^2 \, \mathrm{d}\tau. \tag{51}$$

As in the case of the Lagrangian inclusion the control variable is eliminated from this formulation, resulting in an NLP variable of dimension $4(N+1)$. Note that the running cost remains convex.

4.5. Normal form formulation

A normal form of the formulation of the state dynamics can be easily achieved by solving for q_2 from Eq. (33) in terms of q_1 and $\ddot{q}_1$ and substituting the expression in Eq. (34). Thus, with $w_1 = q_1$, we can write

$$\dot{w}_1 = w_2 \tag{52}$$

$$\dot{w}_2 = w_3 \tag{53}$$

$$\dot{w}_3 = w_4 \tag{54}$$

$$\dot{w}_4 = (\beta_2 \cos w_1 + \beta_3) w_3 + \beta_4 w_2^2 \sin w_1 + \beta_5 \sin w_1 + \beta_1 u \tag{55}$$

where the constants β_i are defined as [22]

$$\beta_1 = \frac{k}{I_1 I_2}, \quad \beta_2 = \frac{-m_1 gl}{I_1}, \quad \beta_3 = \frac{-k(I_1 + I_2)}{I_1 I_2}$$

$$\beta_4 = -\beta_2, \quad \beta_5 = \frac{-m_1 kgl}{I_1 I_2}.$$

Unlike the linear transformation between the state-space formulation and the generalized coordinates (see Eq. (42)), the transformation $\Phi : \boldsymbol{x} \mapsto \mathbf{w}$, from state-space to normal form, is nonlinear and given by

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_3 \\ -\dfrac{m_1 gl \sin x_1 + k(x_1 - x_2)}{I_1} \\ -\dfrac{m_1 gl x_3 \cos x_1 + k(x_3 - x_4)}{I_1} \end{pmatrix}. \tag{56}$$

Because the end-point set for this example is given by two points in the configuration space (including generalized velocities), the nonlinear transformation of Eq. (56) is not detrimental to computation; however, note that if the

problem is slightly altered to require the terminal generalized velocities to lie in a convex set (e.g. $\{0.07 \leq \dot{q}_1 \leq 0.09,\ 0.01 \leq \dot{q}_2 \leq 0.03\}$), Eq. (56) transforms this set to a nonconvex region. In addition, the simple bound constraint representation of the terminal generalized velocities transforms to a representation given by nonlinear functional inequalities.

The size of the NLP variable for this formulation is $5(N+1)$, the same as that for the state-space formulation.

4.6. Normal form inclusion

Formulating the problem as a differential inclusion using the equations in normal form is quite simple and straightforward. The only two differences between this formulation and the normal form are in the rewriting of Eq. (55) as an inequality:

$$\left| \frac{\dot{w}_4 - (\beta_2 \cos w_1 + \beta_3) w_3 - \beta_4 w_2^2 \sin w_1 - \beta_5 \sin w_1}{\beta_1} \right| \leq 15 \tag{57}$$

while the cost function is transformed to

$$J_D = \frac{1}{\beta_1^2} \int_{\tau_0}^{\tau_f} [\dot{w}_4(\tau) - (\beta_2 \cos w_1(\tau) + \beta_3) w_3(\tau) - \beta_4 w_2^2(\tau) \sin w_1(\tau) - \beta_5 \sin w_1(\tau)]^2 \, \mathrm{d}\tau. \tag{58}$$

Obviously, the running cost is now nonconvex in its variables; however, as a result of an elimination of the control variable, the total number of variables in the resulting NLP is now of dimension $4(N+1)$.

4.7. Higher-order differential formulation

Since this system has relative degree 4, the dynamics can be scalarized to a fourth-order system:

$$w_1^{(4)} = \beta_1 u + (\beta_2 \cos w_1 + \beta_3) \ddot{w}_1 + \beta_4 \dot{w}_1^2 \sin w_1 + \beta_5 \sin w_1. \tag{59}$$

The cost function is the same as Eq. (35); i.e. convex. The size of the NLP variable for this formulation is $2(N+1)$.

4.8. Constrained differentially flat formulation

The system is static feedback linearizable and is hence trivially differentially flat with $y = x_1$ as the flat output. It is quite straightforward to show that

$$x_1 = y \tag{60}$$

$$x_2 = \frac{I_1 \ddot{y} + m_1 g l \sin(y)}{k_1} + y \tag{61}$$

$$x_3 = \dot{y} \tag{62}$$

$$x_4 = \frac{I_1 y^{(3)} + m_1 g l \dot{y} \cos(\dot{y})}{k_1} + \dot{y} \tag{63}$$

$$u = \frac{y^{(4)} - (\beta_2 \cos y + \beta_3) \ddot{y} - \beta_4 \dot{y}^2 \sin y - \beta_5 \sin y}{\beta_1}. \tag{64}$$

Since the control is constrained, it is obvious that the flat output is also constrained. Thus, the optimal control problem in flat space is path constrained and can be formulated as finding the function $t \mapsto y$ that minimizes the cost function

$$J_E = \frac{1}{\beta_1^2} \int_{\tau_0}^{\tau_f} (y^{(4)} - (\beta_2 \cos y + \beta_3) \ddot{y} - \beta_4 \dot{y}^2 \sin y - \beta_5 \sin y)^2 \, \mathrm{d}\tau \tag{65}$$

subject to the constraint

$$\frac{1}{\beta_1} |y^{(4)} - (\beta_2 \cos y + \beta_3) \ddot{y} - \beta_4 \dot{y}^2 \sin y - \beta_5 \sin y| \leq 15. \tag{66}$$

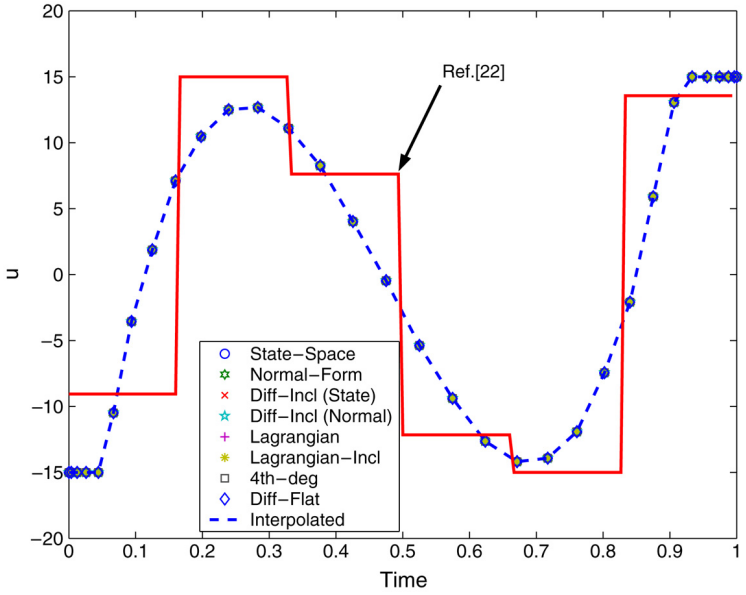

Fig. 2. Control, u, from various methods.

Additional constraints on the problem are the transformed end-point constraints. The end-points are obtained by using the nonlinear transformation provided by Eqs. (60)–(63).

The size of the NLP variable for this formulation is $N + 1$ and it is the smallest among these formulations.

5. Numerical results

The various problem formulations discussed in the previous section were solved using SNOPT [45]. Fig. 2 compares the control profiles obtained by various methods. Clearly, the controls obtained by all of our various formulations for 32 nodes are in agreement with one another but differ from the one obtained in [22]. As shown in Fig. 2, the control of [22] is apparently some piecewise constant approximation to our result. The piecewise constant control is not optimal since our cost of approximately 109 is nearly 30% less than 154, the cost obtained by the piecewise constant control.

For the purpose of completeness, the profiles of the generalized coordinates, q_1 and q_2, are shown in Figs. 3 and 4.

With the exception of the flat formulation, the plots displayed in Figs. 2–4 are representative of the results obtained from 100 runs with random initial guesses. The purpose of the random runs was to eliminate the role played by the initial guess in the computation of the run times. The results reported in Table 1 are averaged over such 100 samples. The run times were based on a MATLAB code with the interface to SNOPT through an executable file available from TOMLAB [46]. The hardware was a Pentium 4, 2.4 GHz PC with 512 MB of RAM.

The first point of note in Table 1 is that almost all formulations take fractions of a second to run, with the higher-order (i.e. fourth-order) formulation capable of producing solutions at a rate of about 30 Hz. The normal form formulation runs a little faster than the standard state-space formulation — apparently exploiting the triangular structure in the LU decomposition. The differential inclusion formulation with the convex cost function runs faster than its nonconvex counterpart (see rows 2 and 3 in Table 1), but slower than the state-space formulations. Apparently, the loss in sparsity in the inclusion formulations cannot be overcome with the decrease in the number of optimization variables, a phenomenon well known in numerical optimization [45]. The fourth, fifth and sixth rows of Table 1 reveal that if the number of variables is sufficiently decreased while maintaining convexity, there is much to be gained with increased performance. Finally, the last row reveals the detrimental effect of destroying convexity for the price of reduction in the number of variables. As the footnotes in Table 1 indicate, the differentially flat formulation had convergence problems with random initial guesses. When a good guess (such as the converged solution of a different formulation) was used as a starting point, the flat formulation did indeed converge to the solution indicated

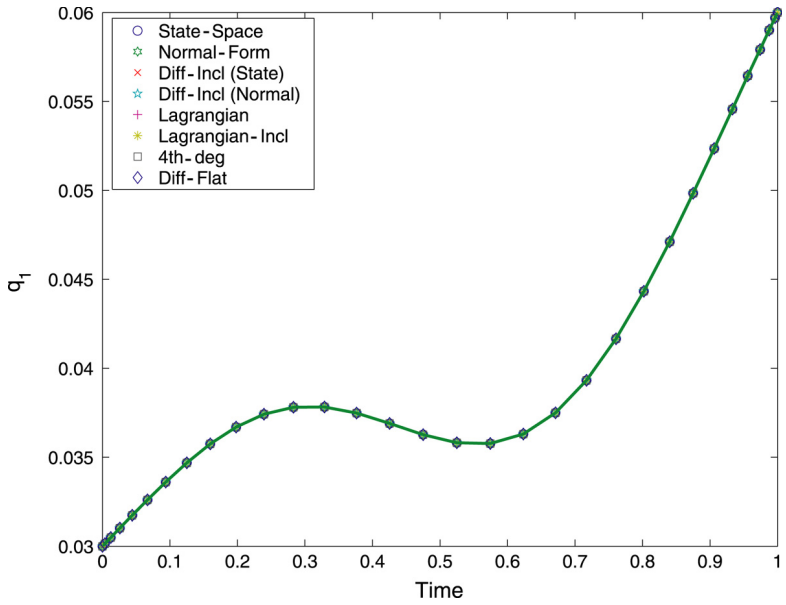

Fig. 3. Generalized coordinate, q_1.

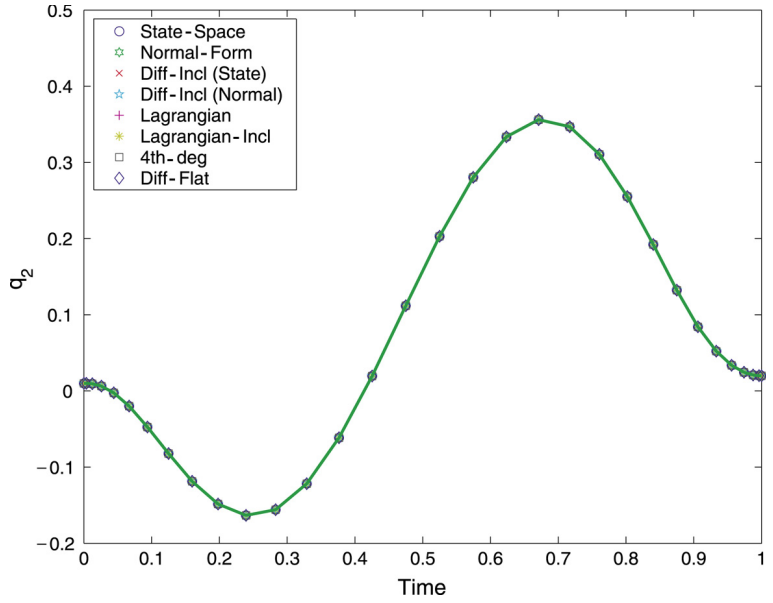

Fig. 4. Generalized coordinate, q_2.

in Figs. 2–4; however, as indicated earlier, the purpose of random guesses was to eliminate the effect of the "closeness" of a guess to the optimal one since the transformations are not isometric.

It is worth noting that if the dense NLP solver, NPSOL [47], is used, the run times are indeed correlated with the number of optimization variables in the sense that the smaller the size of the optimization parameters, the faster the run time. This observation was also reported in [3,4,18,20].

6. Computer implementation issues

Although the run times reported in Table 1 for almost all the formulations are under half a second, several points are worth noting as regards implementation. In many practical situations, it may be an expensive (labor-intensive)

Table 1
Performance of various formulations averaged over 100 random runs

Method	NLP vars	Run time (s)	Cost value	Cost convex
State space	160	0.212	109.6	Y
Normal form	160	0.200	109.6	Y
Diff. incl. (state)	128	0.307	109.6	Y
Diff. incl. (normal)	128	0.335	109.6	N
Euler–Lagrange	96	0.181	109.6	Y
Lagrangian incl.	64	0.111	109.6	Y
Higher order	64	0.034	109.6	Y
Diff. flat	32	16.25[a]	110.7[b]	N

[a] The run time is based on finite-difference computation of Jacobians; see the text for details.
[b] Average based on converged solutions = 92% of the runs.

proposition to explore multiple coordinatizations of a given plant to determine the best one suitable for real-time computation. Motivated by simplifying the implementation of a control algorithm, it is apparent that it is worth exploring implementation methods or ideas that enhance the real-time computation of optimal controls.

In the first place, it is obvious that it is possible to decrease the run time with faster machines and by using software intended for real-time systems. Thus, in the preceding example, we could have achieved even faster run times, across the board, by (a) eliminating all the MATLAB segments of the code, (b) eliminating all the overhead associated with the operating system (Windows), and (c) using a faster computer. In other words, we note that the notion of real-time computation in a modern context has less to do with the computation time achievable on a general purpose desktop computer than the possibilities already offered by existing hardware.

In terms of hardware technology, the fastest achievable run times are by way of application-specific integrated circuits [48] (ASICs). Since ASICs are not re-programmable, field programmable gate arrays (FPGAs) offer computational speeds that are far closer to those of ASICs than those of desktop computers while allowing re-programming, both features of critical value to the computation of control. ASIC and FPGA technologies are widely used in computationally intensive fields such as genome information sciences. In one study [49], the use of FPGAs produced solutions 330 times faster than a Pentium III desktop computer with a 1 GHz processor.

Research on the use of FPGAs for control applications is currently under way at the Naval Postgraduate School. It is clear that the computational speeds already achievable on desktop computers offer real-time computation. What FPGAs offer is the possibility of real-time computation of optimal control on problems that of higher complexity such as those encountered in aerospace applications.

7. Conclusions

Advancements in nonlinear control theory can be quickly exploited by PS methods. One central reason for the versatility of the PS methods is purely geometric: PS methods separate the discretization of the tangent bundle from that of the vector field, and the accuracy of the approximation of the former is largely independent of the latter. Furthermore, higher-order derivatives can be obtained by way of elementary matrix multiplications. Hence the relative degree of a dynamical system can be easily exploited. For similar reasons, differentially flat systems with control constraints or even state-dependent control constraints can be easily handled. Although flatness parameterization appears to be attractive in offering the least number of variables, any loss of convexity resulting from the Lie–Bäcklund isomorphism can be detrimental to real-time computation. There is a significant interplay at the junction where differential-geometric concepts are exploited for computational advantage, the geometric properties of the resulting optimization problem, the preservation of these properties offered by approximation theory, and the numerical properties of large-scale linear algebra. This is an area that promises significant breakthroughs in control theory. It is evident that a combination of PS methods with recent advances in nonlinear control theory and optimization can be combined to effectively solve some challenging control problems.

References

[1] E. Frazzoli, M.A. Dahleh, E. Feron, Real-time motion planning for agile autonomous vehicles, Journal of Guidance, Control and Dynamics 25 (1) (2002) 116–129.

[2] O.A. Yakimenko, I.I. Kaminer, Near optimal trajectory generation for autonomous aircraft landing, in: IEEE Internaltional Symposium on Computer Aided Control System Design, Hawaii, 1999.
[3] M.B. Milam, K. Mushambi, R.M. Murray, A new computational approach to real-time trajectory generation for constrained mechanical systems, in: Proc. IEEE Conf. on Decision and Control, December 2000.
[4] N. Petit, M.B. Milam, R.M. Murray, Inversion based constrained trajectory optimization, in: IFAC-Symp. NOLCOS, 2001.
[5] J. Strizzi, I.M. Ross, F. Fahroo, Towards real-time computation of optimal controls for nonlinear systems, in: Proceedings of the AIAA Guidance, Navigation, and Control Conference, Monterey, CA, 2002. AIAA Paper No. 2002-4945.
[6] I.M. Ross, C.N. D'Souza, Rapid trajectory optimization of multi-agent hybrid systems, in: Proceedings of the AIAA GNC Conference, Providence, RI, 2004.
[7] M. Fliess, J. Lévine, P. Martin, P. Rouchon, Flatness and defect of nonlinear systems: Introductory theory and examples, International Journal of Control 61 (6) (1995) 1327–1361.
[8] M. Fliess, J. Lévine, P. Rouchon, A simplified approach of crane control via a generalized state-space model, in: Proceedings of the 30th Conference on Decision and Control, Brighton, England, 1991.
[9] N. Faiz, S.K. Agrawal, R.M. Murray, Trajectory planning of differentially flat systems with dynamics and inequalities, Journal of Guidance, Control and Dynamics 24 (2) (2001) 219–227.
[10] R.B. Vinter, Optimal Control, Birkhäuser, Boston, MA, 2000.
[11] M. Bardi, I. Capuzzo-Dolcetta, Optimal Control and Viscosity Solutions of Hamilton–Jacobi–Bellman Equations, Birkhäuser, Boston, MA, 1997.
[12] F.H. Clarke, Y.S. Ladyaev, R.J. Stern, P.R. Wolenski, Nonsmooth Analysis and Control Theory, Springer-Verlag, New York, NY, 1998.
[13] M. Fliess, Generalized controller canonical forms for linear and nonlinear dynamics, IEEE Transactions on Automatic Control 35 (1990) 994–1001.
[14] M. Fliess, J. Lévine, P. Martin, P. Rouchon, A Lie–Bäcklund approach to equivalence and flatness of nonlinear systems, IEEE Transactions on Automatic Control 44 (5) (1999) 922–937.
[15] S.K. Agrawal, N. Faiz, A new efficient method for optimization of a class of nonlinear systems without Lagrange multipliers, Journal of Optimization Theory and Applications 97 (1) (1998) 11–28.
[16] J.T. Betts, Practical methods for optimal control using nonlinear programming, in: SIAM: Advances in Control and Design Series, Philadelphia, PA, 2001.
[17] J.T. Betts, Survey of numerical methods for trajectory optimization, Journal of Guidance, Control, and Dynamics 21 (2) (1998) 193–207.
[18] I.M. Ross, F. Fahroo, Pseudospectral methods for optimal motion planning of differentially flat systems, in: Proc. IEEE Conf. on Decision and Control, Las Vegas, NV, December 2002, IEEE Transactions on Automatic Control 49 (8) (2004) 1410–1413.
[19] I.M. Ross, F. Fahroo, A perspective on methods for trajectory optimization, in: Proceedings of the AIAA/AAS Astrodynamics Conference, Monterey, CA, August 2002. AIAA Paper No. 2002-4727.
[20] I.M. Ross, F. Fahroo, A unified computational framework for real-time optimal control, in: Proc. IEEE Conf. on Decision and Control, Maui, HI, 2003.
[21] A. Isidori, Nonlinear Control Systems, Springer-Verlag, New York, NY, 1989.
[22] T. Veeraklaew, S.K. Agrawal, New computational framework for trajectory optimization of higher-order dynamic systems, Journal of Guidance, Control and Dynamics 24 (2) (2001) 228–236.
[23] R.A. Adams, Sobolev Spaces, Academic Press, New York, NY, 1975.
[24] H.J. Sussmann, Some recent results on the maximum principle of optimal control theory, in: C.I. Byrnes, B.N. Datta, D.S. Gilliam, C.F. Martin (Eds.), Systems and Control in the 21st Century, Birkhäuser, Boston, 1997, pp. 351–372.
[25] P.D. Lowen, R.T. Rockafellar, New necessary conditions for the generalized problem of Bolza, SIAM Journal of Control and Optimization 34 (1996) 1496–1511.
[26] I.M. Ross, F. Fahroo, Pseudospectral knotting methods for solving optimal control problems, Journal of Guidance, Control and Dynamics 27 (3) (2004) 397–405.
[27] D.E. Stewart, Rigid-body dynamics with friction and impact, SIAM Review 42 (1) (2000) 3–39.
[28] A.M. Bloch, N.E. Leonard, J.E. Marsden, Controlled Lagrangians and the stabilization of mechanical systems I: The first matching theorem, IEEE Transactions on Automatic Control 45 (2000) 2253–2270.
[29] A.M. Bloch, N.E. Leonard, J.E. Marsden, Controlled Lagrangians and the stabilization of mechanical systems II: Potential shaping, IEEE Transactions on Automatic Control 46 (10) (2001) 1556–1571.
[30] A.M. Bloch, Nonholonomic Mechanics and Control, Springer-Verlag, New York, NY, 2003.
[31] R.T. Rockafellar, Lagrange multipliers and optimality, SIAM Review 35 (1993) 183–283.
[32] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, Cambridge, UK, 2004.
[33] K.E. Brenan, S.L. Campbell, L. Petzold, Numerical solution of initial-value problems in differential-algebraic equations, in: SIAM Series: Classics in Applied Mathematics, Philadelphia, PA, 1996.
[34] B. Fornberg, A Practical Guide to Pseudospectral Methods, Cambridge University Press, New York, NY, 1998.
[35] L.N. Trefethen, Spectral Methods in MATLAB, SIAM, Philadelphia, PA, 2000.
[36] F. Fahroo, I.M. Ross, Second look at approximating differential inclusions, Journal of Guidance, Control and Dynamics 24 (1) (2001) 131–133.
[37] I.M. Ross, J. Rea, F. Fahroo, Exploiting higher-order derivatives in computational optimal control, in: Proc. Med. Conf. Control and Automation, Lisbon, Portugal, 2002.
[38] J. Elnagar, M.A. Kazemi, M. Razzaghi, The pseudospectral Legendre method for discretizing optimal control problems, IEEE Transactions on Automatic Control 40 (10) (1995) 1793–1796.

[39] F. Fahroo, I.M. Ross, Costate estimation by a Legendre pseudospectral method, Journal of Guidance, Control, and Dynamics 24 (2) (2001) 270–277.
[40] F. Fahroo, I.M. Ross, Direct trajectory optimization by a Chebyshev pseudospectral method, Journal of Guidance, Control and Dynamics 25 (1) (2002) 160–166.
[41] B.D. Welfert, Generation of pesudospectral differentiation matrices, SIAM Journal of Numerical Analysis 24 (1997) 1640–1657.
[42] I.M. Ross, F. Fahroo, Legendre pseudospectral approximations of optimal control problems, in: Lecture Notes in Control and Information Sciences, vol. 295, Springer-Verlag, New York, 2003, pp. 327–342.
[43] I.M. Ross, F. Fahroo, Convergence of pseudospectral discretizations of optimal control problems, in: Proc. IEEE Conf. on Decision and Control, Orlando, FL, 2001.
[44] Q. Gong, I.M. Ross, W. Kang, F. Fahroo, Convergence of pseudospectral methods for constrained nonlinear optimal control problems, in: The IASTED International Conference on Intelligent Systems and Control, Honolulu, HI, 2004.
[45] P.E. Gill, W. Murray, M.A. Saunders, SNOPT: An SQP algorithm for large-scale constrained optimization, SIAM Journal of Optimization 12 (4) (2002) 979–1006.
[46] K. Holmström, A.O. Göran, M.M. Edvall, User's guide for Tomlab 4.0.6, Tomlab Optimization, Sweden, August 2003.
[47] P.E. Gill, W. Murray, M.A. Saunders, M.H. Wright, User's guide for NPSOL 5.0: A FORTRAN package for nonlinear programming, Report SOL 86-2, Dept. of Operations Research, Stanford Univ., CA, 1998.
[48] V.P. Nelson, H.T. Nagle, J.D. Irwin, B.D. Carroll, Digital Logic Circuit Analysis and Design, Prentice-Hall, Englewood, NJ, 1995.
[49] Y. Yamaguchi, T. Maruyama, A. Konagaya, An approach for homology search with reconfigurable hardware, Genome Informatics 12 (2001) 374–375.