Sequential Action Control: Closed-Form Optimal Control for Nonlinear and Nonsmooth Systems

Alex Ansari and Todd Murphey

***Abstract*— This paper presents a new model-based algorithm that computes predictive optimal controls on-line and in closed loop for traditionally challenging nonlinear systems. Examples demonstrate the same algorithm controlling hybrid impulsive, underactuated, and constrained systems using only high level models and trajectory goals. Rather than iteratively optimize finite horizon control sequences to minimize an objective, this paper derives a closed-form expression for individual control actions (i.e., control values that can be applied for short duration) that optimally improve a tracking objective over a long time horizon. Under mild assumptions, actions become linear feedback laws near equilibria that permit stability analysis and performance based parameter selection. Globally, optimal actions are guaranteed existence and uniqueness. By sequencing these actions on-line, in receding horizon fashion, the proposed controller provides a min-max constrained response to state that avoids the overhead typically required to impose control constraints. Benchmark examples show the approach can avoid local minima and outperform nonlinear optimal controllers and recent, case-specific methods in terms of tracking performance, and at speeds orders of magnitude faster than traditionally achievable.**

***Index Terms*—real-time optimal control; nonlinear control systems; hybrid systems; impacting systems; closed loop systems.**

I. INTRODUCTION

Robots often present nonlinear, hybrid, underactuated, and high-dimensional constrained control problems. To address these problems, this paper focuses on computational methods that provide a relatively automated policy generation process. The primary contribution is a model-based algorithm that uses optimization to rapidly synthesize predictive optimal controls in real time from a closed-form expression. The process takes advantage of hybrid and underactuated nonlinear dynamics and allows robots to react to the environment on-line. Included examples show this algorithm, which we refer to as *Sequential Action Control* (SAC), can automatically generate controls for traditionally challenging classes of (nonlinear) robotic systems (e.g., continuous, underactuated, hybrid, impulsive, and constrained systems) using only robot models and trajectory goals.

Figure 1 provides an overview of the proposed SAC process. Each cycle computes an optimal *action* (see Def. 1 and the blue shaded bar in Fig. 1) that is sent to a robot. At fixed sampling times, SAC incorporates feedback and repeats the cycle to compute and apply the next action. Subsequent actions are usually synthesized and applied before the robot completes implementing the previously planned control. Although each action is designed for a short application duration, they each improve system performance for a relatively long horizon (from $[t_0, t_f]$ in Fig. 2) from the (receding) current time.

Definition 1. *Actions are defined by the triplet consisting of a control's value, $u \in \mathbb{R}^m$, application duration, $\lambda \in \mathbb{R}^+$, and application time, $\tau \in \mathbb{R}$.*

Procedurally, SAC is similar to continuous-time, nonlinear receding horizon control. That is, SAC constructs a closed-loop control response from a succession of open-loop, finite horizon optimal control problems. However, receding horizon implementations seek control curves that minimize the trajectory objective over each (receding) horizon. For nonlinear systems, each of these optimal control problems is non-convex, requiring an expensive process of iterative optimization that limits implementation to low bandwidth and is subject to local minima [2], [3].

Instead of searching for control curves that directly minimize a trajectory objective, each cycle of SAC computes an *infinitesimal* (duration) action that would optimally *improve* performance over the current horizon. Assuming actions are applied infinitesimally, the process does not need to enforce nonlinear dynamic constraints. As a result, SAC computes an entire *schedule* (curve) providing the optimal infinitesimal action at every time over the current horizon from a convex objective. Section II shows the convex objective yields closed-form solutions with guaranteed optimality, existence, and uniqueness. To choose a "best" action, SAC searches the action schedule for an optimal time to act over the current horizon and uses a line search to generate a short (finite) duration that guarantees long time horizon improvement in trajectory. Benchmark examples show SAC outperforming case-specific methods and popular optimal control algorithms (sequential quadratic programming [77] and iLQG [67]). Compared to these alternatives, SAC computes high-bandwidth (1 KHz) closed-loop trajectories with equivalent or better final cost in significantly less time (milliseconds/seconds vs. hours).

By optimizing over the space of individual (infinitesimal) control actions, Section II-D (and the Appendix) shows SAC requires none of the traditional overhead to deal with control saturation constraints. Additionally, SAC does not need to solve the challenging two-point boundary-value problems used to derive finite horizon optimal controls with indirect methods, nor does it discretize to solve the high dimensional nonlinear programming problems faced by direct methods [54]. Instead, SAC avoids iterative optimization and makes constrained optimal control calculation roughly as inexpensive as simulation. Thus measurement incorporation and feedback synthesis can occur at higher bandwidth. Because controls can be expressed as an analytical formula, the method is easily implemented in code. In short, SAC enables predictive optimal control

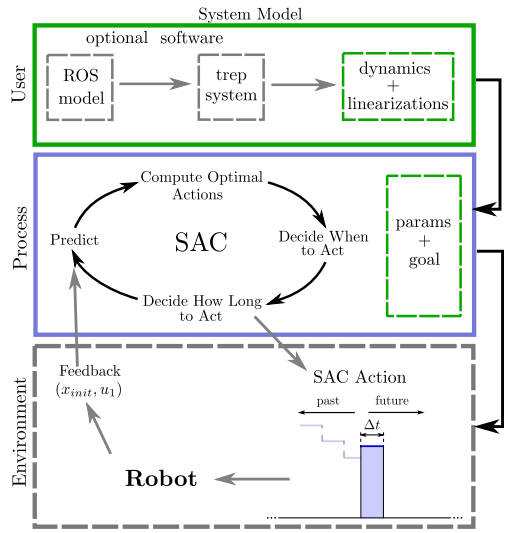


Fig. 1: An overview of the SAC control process including possible open-source interfaces (e.g., ROS [52] and trep [32]).

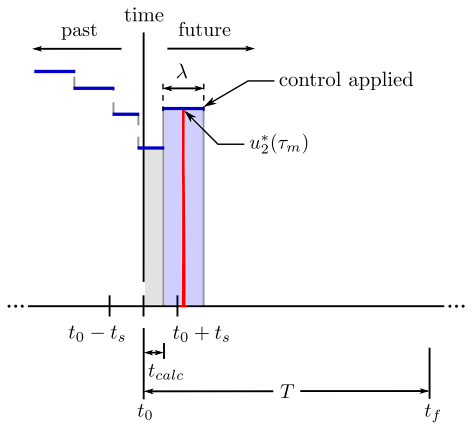


Fig. 2: Following the cyclic process in Fig. 1, SAC computes a *schedule*, $u_2^* : [t_0, t_f] \mapsto \mathbb{R}^m$, providing the value of optimal infinitesimal actions that maximally improve a tracking objective over the current (moving) horizon. Next, SAC selects an application time, $\tau_m \in (t_0 + t_{calc}, t_f)$, and the value of the resulting infinitesimal optimal action (in red) specifies the value of the next SAC action (blue shaded bar). A line search sets the action's duration, λ. Previously computed actions are applied while current calculations complete, $t \in [t_0, t_0 + t_{calc}]$.

methods to be applied on-line and in closed-loop for robotic systems where such methods would normally prove infeasible.

This paper is divided into two parts. Part I (Sections II - III) derives SAC for a broad class of differentiable nonlinear control-affine systems. Additionally, Section II includes stability results and conditions under which optimal actions become locally linear feedback laws that facilitate parameter selection. The section also derives efficient means to impose min-max control constraints, demonstrated in the examples in Section III. Part II (Sections IV-VI) shows that extending SAC to control of hybrid impulsive systems only requires modification a single differential equation governing an adjoint variable. Section V-C presents illustrative examples, including on-line control of a 3D spring-loaded inverted pendulum up a flight of stairs. Conclusions and future work are in Section VI. Table I includes notation used throughout this paper.

TABLE I: Notation

symbol	description
$D_x f(\cdot)$	partial derivative $\frac{\partial f(\cdot)}{\partial x}$
$\|\cdot\|_M$	norm where M provides the metric
	(e.g., $\|x(t)\|_Q^2 = x(t)^T Q\, x(t)$)
R^{-T}	equivalent to $(R^T)^{-1}$
$R > 0$	indicates R is positive definite ($\geq$ for semi-definite)

Part I: SAC for Differentiable Systems

II. Control Synthesis

A. Analytical Solution for Infinitesimal Optimal Actions

This section presents a method to compute the schedule of infinitesimal optimal actions associated with the current horizon $T = [t_0, t_f]$ based on early work in [5]. Calculations result in a closed-form expression for this schedule for systems with dynamics,

$$\dot{x}(t) = f(t, x(t), u(t)) \tag{1}$$

nonlinear in state $x : \mathbb{R} \mapsto \mathbb{R}^{n\times 1}$. Though these methods apply more broadly, the optimal action schedule is derived for the case where (1) is linear with respect to the control, $u : \mathbb{R} \mapsto \mathbb{R}^{m\times 1}$, satisfying control-affine form,

$$f(t, x(t), u(t)) = g(t, x(t)) + h(t, x(t))\, u(t)\,. \tag{2}$$

The cost functional,

$$J_1 = \int_{t_0}^{t_f} l_1(t, x(t))\, dt + m(x(t_f))\,, \tag{3}$$

measures performance to gauge the improvement provided by optimal actions.[1] For brevity, the time dependence in (1), (2), and (3) will be dropped. The following definitions and assumptions clarify the systems and cost functionals addressed.

Definition 2. *Piecewise continuous functions will be referred to as $\widetilde{\mathcal{C}}^0$. These functions will be defined according to one of their one-sided limits at discontinuities.*

Definition 3. *Though actions are defined by a triplet consisting of a value, duration, and application time (see Def. 1), for the sake of brevity, we will refer to actions according to their value and will disambiguate by specifying the application time and duration as required. As such, $\mathbb{U}_2 \triangleq \left\{ u_2^*(\tau_{m,i}) \,\middle|\, i \in \{1, \ldots, c\} \right\}$ will represent the set of optimal actions applied by SAC with $\tau_{m,i}$ as an action's application time. The maximal index $c \in \mathbb{N}$ corresponds to the last action applied. The set of application intervals associated with each action is $\Upsilon \triangleq \left\{ \Upsilon_i \right\}_{i\in\{1,\ldots c\}}$, such that $\Upsilon_i = [\tau_{m,i} - \frac{\lambda_i}{2}, \tau_{m,i} + \frac{\lambda_i}{2}]$ and λ_i is an action's duration.*

Assumption 1. *The elements of dynamics vector* (1) *are real, bounded, $\mathcal{C}^1$ in x, and $\mathcal{C}^0$ in t and u.*

[1] Though not required, (3) should be non-negative if it is to provide a performance measure in the formal sense.

Assumption 2. *The terminal cost, $m(x(t_f))$, is real and differentiable with respect to $x(t_f)$. Incremental cost $l_1(x)$ is real, Lebesgue integrable, and $\mathcal{C}^1$ in x.*

Assumption 3. *SAC control signals, u, are real, bounded, and $\widetilde{\mathcal{C}}^0$ such that*

$$u(t) = \begin{cases} u_1(t) & : t \notin \Upsilon_i \\ u_2^*(\tau_{m,i}) & : t \in \Upsilon_i \end{cases},$$

with nominal control, u_1, that is $\mathcal{C}^0$ in t.[2]

The planning process assumes the system switches from a (default) nominal mode,

$$f_1(t) \triangleq f(x(t), u_1(t)),$$

to the mode associated with the optimal action,

$$f_2(t, \tau_{m,i}) \triangleq f(x(t), u_2^*(\tau_{m,i})),$$

and then back to f_1 over the course of each horizon $[t_0, t_f]$.

Consider the case where the system evolves according to nominal control dynamics f_1, and optimal action $u_2^*(\tau_{m,i})$ is applied (dynamics switch to f_2) for an infinitesimal duration before switching back to nominal control. This is the condition depicted by the red curve in Fig. 2 where nominal control $u_1 = 0$. In this case, the *mode insertion gradient* [15], [16],

$$\frac{dJ_1}{d\lambda_i^+} = \rho(s)^T (f_2(s,s) - f_1(s)) \quad \forall s \in [t_0, t_f], \tag{4}$$

evaluated at $s = \tau_{m,i}$ measures the first-order sensitivity of the cost (3) to infinitesimal application of f_2. Note (4) assumes the state in f_1 and f_2 is defined from the nominal control, $x(s) \triangleq x(s, u_1(s))\ \forall s \in [t_0, t_f]$, $\rho : \mathbb{R} \mapsto \mathbb{R}^{n\times 1}$ is the adjoint (co-state) variable calculated from the nominal trajectory,[3]

$$\dot{\rho} = -D_x l_1(x)^T - D_x f_1(x, u_1)^T \rho, \tag{5}$$

with $\rho(t_f) = D_x\, m(x(t_f))^T$, and the control duration is evaluated infinitesimally, as $\lambda_i \to 0^+$.

In hybrid systems literature, the mode insertion gradient provides the first-order sensitivity of a cost function to an infinitesimal duration switch in dynamics. The term is used in mode scheduling [12], [16], [72], [73], to determine the optimal time to insert control modes assuming the modes are known a-priori. This work demonstrates how it can be used to solve for the value of infinitesimal optimal actions (new modes) at each instant. A thorough discussion and a general derivation that extends (4) to hybrid impulsive dynamical systems with resets and objectives that depend on the control is included in Section IV.

While infinitesimal optimal actions cannot change the state or tracking cost, the cost (3) is sensitive to their application. The mode insertion gradient (4) indicates this sensitivity at any potential application time, $s \in [t_0, t_f]$. To specify a desired sensitivity, a control objective selects infinitesimal optimal actions that drive (4) to a desired negative value, $\alpha_d \in \mathbb{R}^-$. At any potential application time $s \in [t_0, t_f]$, the infinitesimal optimal action that minimizes,

$$\begin{aligned} l_2(s) &\triangleq l_2(x(s), u_1(s), u_2(s), \rho(s)) \\ &= \frac{1}{2}[\frac{dJ_1}{d\lambda_i^+} - \alpha_d]^2 + \frac{1}{2}\|u_2(s)\|_R^2 \\ &= \frac{1}{2}[\rho(s)^T (f_2(s,s) - f_1(s)) - \alpha_d]^2 + \frac{1}{2}\|u_2(s)\|_R^2, \end{aligned} \tag{6}$$

minimizes control authority in achieving the desired sensitivity. The first expression in (6) highlights that the objective corresponds to actions at a specific time. The matrix $R = R^T > 0$ provides a metric on control effort. Because the space of positive semi-definite / definite cones is convex (see [11]), (6) is convex with respect to infinitesimal actions $u_2(s)$.

To minimize (6) and return the infinitesimal optimal action at any potential application time $s \in [t_0, t_f]$, the mode insertion gradient must exist at that time. This requires continuity (in time) of ρ, f_1, f_2, and u_1 in a neighborhood of s (see [15]). While Assumps. 1-3 ensure continuity requirements are met, the assumptions are overly restrictive, and the results of this paper can be generalized.

With Assumps. 1-3, the mode insertion gradient exists, is bounded, and (6) can be minimized with respect to $u_2(s)\ \forall s \in [t_0, t_f]$. The following theorem finds this minimum to compute the schedule of infinitesimal optimal actions.

Theorem 1. *Define $\Lambda \triangleq h(x)^T \rho\, \rho^T h(x)$. The schedule of infinitesimal optimal actions,*

$$u_2^*(t) \triangleq \arg\min_{u_2(t)} \ l_2(t) \qquad \forall t \in [t_0, t_f], \tag{7}$$

to which cost (3) *is optimally sensitive at any time is*

$$u_2^* = (\Lambda + R^T)^{-1} [\Lambda\, u_1 + h(x)^T \rho\, \alpha_d]. \tag{8}$$

Proof: Evaluated at any time $t \in [t_0, t_f]$, the schedule of infinitesimal optimal actions, u_2^*, provides an infinitesimal optimal action that minimizes (6) at that time. The schedule therefore also minimizes the (infinite) sum of costs (6) associated with the infinitesimal optimal action at every time $\forall t \in [t_0, t_f]$. Hence, (7) can be obtained by minimizing

$$J_2 = \int_{t_0}^{t_f} l_2(x(t), u_1(t), u_2(t), \rho(t))\, dt. \tag{9}$$

Because the sum of convex functions is convex, and x in (4) is defined only in terms of u_1, minimizing (9) with respect to $u_2(t)\ \forall t \in [t_0, t_f]$ is convex and unconstrained. It is necessary and sufficient for (global) optimality to find the u_2^* for which the first variation of (9) is $0\ \forall\, \delta u_2^* \in \mathcal{C}^0$. Using the Gâteaux derivative and the definition of the functional derivative,

$$\begin{aligned} \delta J_2 &= \int_{t_0}^{t_f} \frac{\delta J_2}{\delta u_2^*(t)} \delta u_2^*(t)\, dt \\ &= \frac{d}{d\epsilon} \int_{t_0}^{t_f} l_2(x(t), u_1(t), u_2^*(t) + \epsilon\, \eta(t), \rho(t))\, dt|_{\epsilon=0} \\ &= \int_{t_0}^{t_f} \frac{d}{d\epsilon} l_2(x(t), u_1(t), u_2^*(t) + \epsilon\, \eta(t), \rho(t))|_{\epsilon=0}\, dt \\ &= \int_{t_0}^{t_f} \frac{\partial l_2(x(t), u_1(t), u_2^*(t), \rho(t))}{\partial u_2(t)}\, \eta(t)\, dt \\ &= 0, \end{aligned} \tag{10}$$

[2]The dynamics and nominal control can be $\widetilde{\mathcal{C}}^0$ in t if application times $\tau_{m,i}$ exclude points of discontinuity in $u_1(t)$.

[3]As opposed to traditional fixed-horizon optimal control methods [41], [50], this adjoint is easily computed because it does not depend on the closed-loop, optimal state $x^*(t, u_2^*(\tau_{m,i}))$.

where ϵ is a scalar and $\epsilon\,\eta = \delta u_2^*$.

The final equivalence in (10) must hold $\forall\,\eta$. A generalization of the Fundamental Lemma of Variational Calculus (see [47]), implies $\frac{\partial l_2(\cdot)}{\partial u_2} = 0$ at the optimizer. The resulting expression,

$$\begin{aligned}\frac{\partial l_2(\cdot)}{\partial u_2}^T &= \left[(\rho^T h(x)\,[u_2^* - u_1] - \alpha_d)\rho^T h(x) + u_2^{*T} R\right]^T \\ &= h(x)^T \rho\,(\rho^T h(x)[u_2^* - u_1] - \alpha_d) + R^T u_2^* \\ &= [h(x)^T \rho\,\rho^T h(x)]\,u_2^* + R^T u_2^* \\ &\quad -[h(x)^T \rho\,\rho^T h(x)]u_1 - h(x)^T \rho\,\alpha_d \\ &= 0\,, \end{aligned} \tag{11}$$

can therefore be solved in terms of u_2^* to find the schedule that minimizes (6) at any time. Algebraic manipulation confirms this optimal schedule is (8). ∎

In searching for finite horizon optimal control curves that directly minimize an objective, the dynamical constraints in nonlinear optimal control produce non-convex optimization problems (even when the objective is convex). Such problems require iterative optimization, often performed by linearizing dynamics and quadratizing cost about the current trajectory and solving a succession of convex sub-problems [26], [66], [69]. The process is costly and solutions are still locally optimal with respect to the constrained objective (subject to local minima). In contrast, SAC searches for actions to which (3) is maximally sensitive using a separate, unconstrained objective (9). In addition to providing a closed-form solution for the entire schedule of optimal actions (8), the following corollary shows (8) also inherits powerful guarantees and is a global optimizer *even for non-convex tracking objectives* (3).

Corollary 1. *Solutions* (8) *exist, are unique, and globally optimize* (9)*.*

Proof: The proof follows from the fact that minimization of (9) is convex with a continuous first variation (10) (based on Assumps. 1-3). These conditions guarantee the existence and uniqueness of solutions (8) that cause (10) to vanish locally, which is both necessary and sufficient for global optimality. ∎

In addition to the properties of Corollary 1, near equilibrium points, solutions (8) simplify to linear state feedback laws. This linear form permits local stability analysis (and parameter selection) based on continuous systems techniques.

Corollary 2. *Assume system* (2) *contains equilibrium point* $x = 0$*, the state tracking cost* (3) *is quadratic,*[4]

$$J_1 = \frac{1}{2}\int_{t_0}^{t_f} \|x(t) - x_d(t)\|_Q^2\,dt + \frac{1}{2}\,\|x(t_f) - x_d(t_f)\|_{P_1}^2\,, \tag{12}$$

with $x_d = x_d(t_f) = 0$*,* $Q = Q^T \geq 0$*, and* $P_1 = P_1{}^T \geq 0$ *defining measures on state error, and* $u_1 = 0$*. There exists neighborhoods around the equilibrium,* $\mathcal{N}(0)$*, and final time,* $\mathcal{N}(t_f)$*, where optimal actions* (8) *are equivalent to linear feedback regulators,*

$$u_2^*(t) = R^{-1} h(0)^T\,P(t)\,x(t)\,\alpha_d \quad \forall t \in \mathcal{N}(t_f)\,. \tag{13}$$

[4]Quadratic cost (12) is assumed so that resulting equations emphasize the local similarity between SAC controls and LQR [4].

Proof: At final time, $\rho(t_f) = P_1 x(t_f)$. Due to continuity Assumps. 1-2, this linear relationship between the state and adjoint must exist for a nonzero neighborhood of the final time, $\mathcal{N}(t_f)$, such that

$$\rho(t) = P(t)\,x(t) \quad \forall t \in \mathcal{N}(t_f)\,. \tag{14}$$

Applying this relationship, (8) can formulated as

$$\begin{aligned} u_2^* = &(h(x)^T P\,x\,x^T P^T h(x) + R^T)^{-1} \\ &[h(x)^T P\,x\,x^T P^T h(x)\,u_1 + h(x)^T P\,x\,\alpha_d]\,. \end{aligned}$$

This expression contains terms quadratic in x. For $x \in \mathcal{N}(0)$, these quadratic terms go to zero faster than the linear terms, and controls converge to (13).

By continuity Assump. 1 and for $x \in \mathcal{N}(0)$, the dynamics (1) can be approximated as an LTI system, $\dot{x} = A\,x + B\,u$, (where A and B are linearizations about the equilibrium). Applying this assumption and differentiating (14) produces

$$\begin{aligned}\dot{\rho} &= \dot{P}\,x + P\,\dot{x} \\ &= \dot{P}\,x + P\,(A\,x + B\,u_1)\,. \end{aligned}$$

Inserting relation (5) yields

$$-D_x l_1(x)^T - A^T P\,x = \dot{P}\,x + P(A\,x + B\,u_1)\,,$$

which can be re-arranged such that

$$0 = (Q + \dot{P} + A^T P + P\,A)\,x + P\,B\,u_1.$$

When the nominal control $u_1 = 0$, this reduces to

$$0 = Q + A^T P + P\,A + \dot{P}\,. \tag{15}$$

Note the similarity to a Lyapunov equation. As mentioned, this relationship must exist for a nonzero neighborhood, $\mathcal{N}(t_f)$. Therefore, there exists neighborhoods $\mathcal{N}(t_f)$ and $\mathcal{N}(0)$ in which schedule (8) simplifies to a time-varying schedule of linear feedback regulators (13),[5] where $P(t)$ can be computed from (15) subject to $P(t_f) = P_1$. ∎

For the assumptions in Corollary 2, actions (13) are linear time-varying state feedback laws near equilibrium that can be used to assess closed-loop stability. Although the corollary is derived in the neighborhood $\mathcal{N}(t_f)$, because (15) is linear in P, (15) cannot exhibit finite escape time. Through a global version of the Picard–Lindelöf theorem [34], it is straightforward to verify (15) (and therefore (13)) exists and is unique for arbitrary horizons and not only for $t \in \mathcal{N}(t_f)$. Assuming the horizon, T, is fixed and SAC continuously applies actions at the (receding) initial time, $t = t_0$, (13) yields a *constant* feedback law, $u_2^*(t) = -K\,x(t)$, where K depends on the system's linearizations, weight matrices, Q, R, and P_1, the time horizon, T, and the α_d term.[6] Thus LTI stability conditions may be applied to facilitate parameter selection. Similarly, one can also show Corollary 2 yields a feedback expression in error coordinates for which LTV

[5]Note the $h(0)^T = B^T$ term in (13) shows up because the system is assumed to be in a neighborhood where the dynamics can be linearly modeled.

[6]Sums-of-Squares (SOS) and the S-procedure [49], [68] can pre-compute regions of attraction for (13) and so can determine when SAC should switch to continuous application of (13).

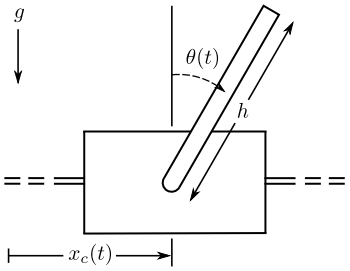


Fig. 3: Configuration variables for the cart-pendulum system.

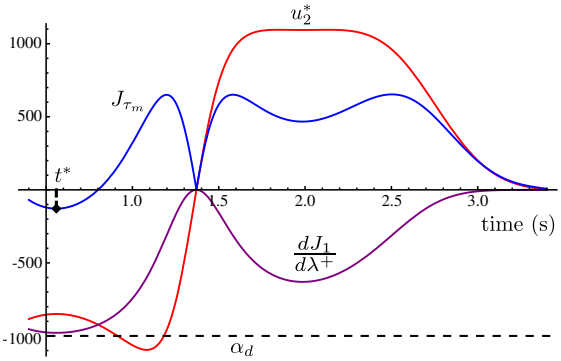


Fig. 4: A schedule of optimal actions, u_2^*, is depicted (red curve) for a $T = 3\,s$ predicted trajectory of the cart-pendulum system (16) starting at current time $t_0 = 0.4\,s$. These actions minimize acceleration in driving the mode insertion gradient toward $\alpha_d = -1,000$. The (purple) mode insertion gradient curve approximates the change in cost (3) achievable by short application of $u_2^*(t)$ at different points in time. The objective, J_{τ_m}, (blue curve) is minimized to find an optimal time, t^*, to act. Waiting to act at $\tau_{m,i} = t^*$ rather than at $\tau_{m,i} = t_0$, SAC generates greater cost reduction using less effort.

stability analysis can be used to identify parameters that guarantee local stability to a desired trajectory, $x_d(t)$.

As a final point, if (3) is quadratic and the nominal control, u_1, modeled as applying consecutively computed optimal actions (13) near equilibrium, (15) becomes a Riccati differential equation for the closed-loop system (see [27]) and actions (13) simplify to finite horizon LQR controls [4]. In this case one can prove the existence of a Lyapunov function ((15) with $\dot{P} = 0$) and guarantee stability for SAC using methods from LQR theory [27] to drive $\dot{P} \to 0$. As for receding horizon control, Lyapunov functions can be constructed using infinite horizons or a terminal cost and constraints that approximate the infinite horizon cost [2], [14], [25], [29], [38], [44].

B. Computing the Time for Control Application

Theorem 1 provides a schedule of infinitesimal optimal control actions that minimize control authority while tracking a desired sensitivity, α_d, in cost (3) at any application time $\tau_{m,i} \in [t_0, t_f]$. By continually computing and applying optimal actions $u_2^*(\tau_{m,i})$ immediately (at $t_0 + t_{calc}$), SAC can approximate a continuous response. However, it is not always desirable to act at the beginning of each horizon.

Consider a simple model of a cart-pendulum where the state vector consists of the angular position and velocity of the pendulum and the position and lateral velocity of the cart, $x = (\theta,\, \dot{\theta},\, x_c,\, \dot{x}_c)$. With the cart under acceleration control, $u = (a_c)$, the underactuated system dynamics are modeled as

$$f(x,u) = \begin{pmatrix} \dot{\theta} \\ \frac{g}{h}\sin(\theta) + \frac{a_c\,\cos(\theta)}{h} \\ \dot{x}_c \\ a_c \end{pmatrix}. \tag{16}$$

The constant h represents the pendulum length, specified as 2 m, and g is the gravitational constant. Figure 3 depicts the relevant configuration variables. Note that any time the pendulum is horizontal (e.g., $\theta(t) = \frac{\pi}{2}$ rad.), no action can push θ toward the origin to invert the pendulum.

To find the most effective time $\tau_{m,i} \in [t_0 + t_{calc}, t_f]$ to apply a control action from u_2^*, SAC searches for application times that optimize an objective function,

$$J_{\tau_m}(t) = \|u_2^*(t)\| + \left.\frac{dJ_1}{d\lambda_i^+}\right|_t + (t - t_0)^{\beta}, \tag{17}$$

defining the trade-off between control efficiency and the cost of waiting.[7]

Fig. 4 shows the schedule of optimal actions, u_2^*, computed for (16) starting at $t_0 = 0.4\,s$ into an example closed-loop SAC trajectory. Actions in u_2^* are designed to drive the mode insertion gradient (purple curve), which is proportional to the achievable change in (3), toward $\alpha_d = -1,000$ if switched to at any time. At $t \approx 1.39\,s$ the pendulum passes through the horizontal singularity and control becomes ineffective, as indicated by the mode insertion gradient going to 0. The mode insertion gradient also goes to 0 toward the end of the trajectory since no finite control action can improve (3) if applied at the final time. As the curve of J_{τ_m} vs time (blue) indicates, to optimize the trade-off between wait time and effectiveness of the action, the system should do nothing and drift until optimal (as defined by (17)) time $t^* \approx 0.57\,s$.

C. Computing the Control Duration

Temporal continuity of ρ, f_1, f_2 and u_1 provided by Assump. 1-3 ensures the mode insertion gradient (sensitivity of the tracking cost) is continuous with respect to duration around where $\lambda_i \to 0^+ \;\forall\; \tau_{m,i} \in [t_0, t_f]$. Therefore, there exists an open, non-zero neighborhood, $V_i = \mathcal{N}(\lambda_i \to 0^+)$, where the sensitivity indicated by (4) models the change in cost relative to application duration to first-order (see [12], [16] and the generalized derivation in Section IV). For finite durations, $\lambda_i \in V_i$, the change in tracking cost (3) is locally modeled as

$$\Delta J_1 \approx \left.\frac{dJ_1}{d\lambda_i^+}\right|_{s\,=\,\tau_{m,i}} \lambda_i. \tag{18}$$

[7] Implementation examples apply $\beta = 1.6$ as a balance between time and control effort in achieving tracking tasks, but any choice of $\beta > 0$ will work.

As $u_2^*(\tau_{m,i})$ regulates $\frac{dJ_1}{d\lambda_i^+} \approx \alpha_d$, (18) becomes $\Delta J_1 \approx \alpha_d \lambda_i$. Thus the choice of λ_i and α_d allows the control designer to specify the desired degree of change provided by actions, $u_2^*(\tau_{m,i})$. We use a line search with a simple descent condition to find a $\lambda_i \in V_i$ that yields the desired change [77].

Because the pair (α_d, λ_i) determines the change in cost each action can provide, it is worth noting that a sufficient decrease condition similar to the one proposed in [12] can be applied to the choice of λ_i.

D. Input Saturation

As a benefit of SAC, actions computed from (8) can be saturated to satisfy min-max constraints using 1) quadratic programming, 2) by scaling the control vector, or 3) by scaling components of the control vector. The Appendix includes an analysis and proves each method for the case where $u_1 = 0$, as in all the examples in this paper. All the examples to follow use the third option and replace elements of the control with saturated versions. The approach avoids all computational overhead and produces constrained optimal actions that guarantee improvement in cost. For an overview of the SAC approach outlining the on-line procedure for synthesis of constrained optimal actions, selection of actuation times, and resolution of control durations, refer to Algorithm 1.

III. Example Systems

The following section provides simulation examples that apply SAC on-line in benchmark underactuated control tasks.[8] Each example emphasizes a different performance related aspect of SAC and results are compared to alternative methods.

A. Cart-Pendulum

First, we present 3 examples where SAC is applied to the nonlinear cart-pendulum (16) in simulated constrained swing-up. Performance is demonstrated using the cart-pendulum as it provides a well understood underactuated control problem that has long served as a benchmark for new control methodologies (see [7], [10], [42], [46], [63], [80]).

1) Low-Frequency Constrained Inversion: This example uses SAC to invert the cart-pendulum (16) with low frequency (10 Hz) feedback and control action sequencing to highlight the control synthesis process. Control constraints, $\ddot{x}_c \in [-4.8, 4.8]\,\frac{m}{s^2}$, show SAC can find solutions that require multiple swings to invert. We use a quadratic tracking cost (12) with the state dependent weights, $Q(x(t)) = Diag[\,200, 0, (x_c(t)/2)^8, 50\,]$. to impose a barrier / penalty function (see [11], [71]) that constrains the cart's state so $x_c \in [-2, 2\,]$. Terminal and control costs in (6) and (12) are defined using $P_1 = 0$, $R = [0.3]$, and a horizon of $T = 1.5\,\text{s}$.[9]

Results in Fig. 5 correspond to an initial condition with the pendulum hanging at the stable equilibrium and zero initial velocity, $x_{init} = (\pi, 0)$. The red curve shows the penalty function successfully keeps the cart position within $[-2, 2\,]\,\text{m}$. The simulated trajectory is included in the video attachment.

[8] We also have trajectory tracking results (e.g, for differential drive robots) but cannot include them due to space constraints.

[9] All examples use wrapped angles $\in [-\pi, \pi)$ rad.

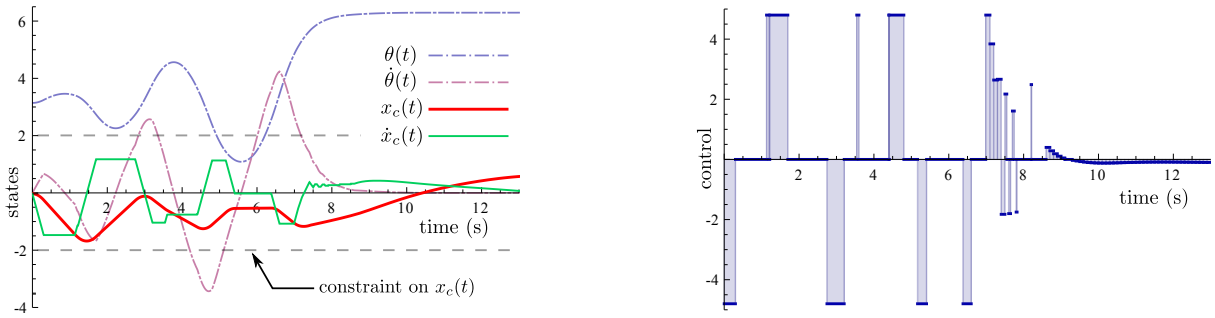
Algorithm 1 Sequential Action Control

Initialize α_d, minimum change in cost ΔJ_{min}, current time t_{curr}, default control duration Δt_{init}, nominal control u_1, scale factor $\omega \in (0, 1)$, prediction horizon T, sampling time t_s, the max time for iterative control calculations t_{calc}, the max backtracking iterations k_{max}, and action iteration i.
while $t_{curr} < \infty$ **do**
  $i = i + 1$
  $(t_0, t_f) = (t_{curr}, t_{curr} + T)$
  Simulate (x, ρ) from f_1 for $t \in [t_0, t_f]$
  Compute initial cost $J_{1,init}$
  Specify α_d
  Compute u_2^* from (x, ρ) using Theorem 1
  Specify / search for time, $\tau_{m,i} > t_0 + t_{calc}$, to apply u_2^*
  Saturate $u_2^*(\tau_{m,i})$
  Initialize $k = 0$, $J_{1,new} = \infty$
  while $J_{1,new} - J_{1,init} > \Delta J_{min}$ **and** $k \leq k_{max}$ **do**
    $\lambda_i = \omega^k \Delta t_{init}$
    $(\tau_0, \tau_f) = (\tau_{m,i} - \frac{\lambda_i}{2}, \tau_{m,i} + \frac{\lambda_i}{2})$
    Re-simulate x applying f_2 for $t \in [\tau_0, \tau_f]$
    Compute new cost $J_{1,new}$
    $k = k + 1$
  end while
  $u_1(t) = u_2^*(\tau_{m,i})\ \forall t \in [\tau_0, \tau_f] \cap [t_0 + t_s, t_0 + t_s + t_{calc}]$
  while $t_{curr} < t_0 + t_s$ **do**
    Wait( )
  end while
end while

Algorithm 1: At sampling intervals SAC incorporates feedback and simulates the system with a nominal (typically null) control. Optimal alternative actions are computed as a closed-form function of time. A time is chosen to apply the control action. A line search provides a duration that reduces cost.

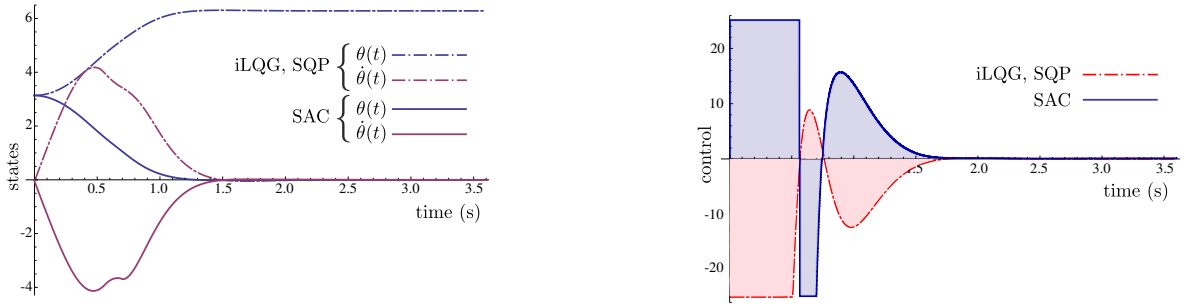
2) High-Frequency Constrained Inversion: In this example, SAC performs on-line swing-up and cart-pendulum inversion with high-frequency feedback (1 KHz). To gauge the quality of the inversion strategy, we compare the on-line, closed-loop SAC control to the off-line solution from trajectory optimization using MATLAB's sequential quadratic programming (SQP) and iLQG implementations. The SQP method is widely used and underlies the approach to optimization in [17], [21], [31], [39], [55], [56]. The iLQG [66], [69] algorithm is a state-of-the-art variant of differential dynamic programming (DDP). While early versions did not accommodate control constraints, iLQG achieves a 10 fold speed improvement over DDP in simulations [40] and has since been applied for real-time humanoid control [66]. This section compares to a recent variant that incorporates control constraints through a new active-set method [67]. We use a publicly available MATLAB iLQG implementation developed by its authors.[10]

[10] Available at http://www.mathworks.com/matlabcentral/fileexchange/52069-ilqg-ddp-trajectory-optimization.



(a) (b)

Fig. 5: SAC inverts the cart-pendulum at a low sampling and control sequencing frequency of 10 Hz (at equilibrium the dynamics correspond to a simple pendulum with natural frequency of 0.35 Hz). This low-frequency control signal (Fig. 5b) illustrates how individual actions are sequenced (especially apparent from 7 to 10 s). SAC maintains the cart in $[-2, 2]$ m during inversion. Figure 5b also shows SAC automatically develops an energy pumping strategy to invert the pendulum.



(a) (b)

Fig. 6: SAC can provide control solutions on-line and in closed-loop (these results reflect $1,000$ Hz feedback) that achieve performance comparable to or better than solutions from nonlinear optimal control. For the trajectory depicted, SAC achieves the same final cost of $J_{pend} \approx 2,215$ as SQP and iLQG.

To highlight the sensitivity of optimal control (i.e., iLQG and SQP) to local equilibria even on simple nonlinear problems (and to speed SQP computations), this example uses a low-dimensional cart-pendulum model. The simplified model leaves the cart position and velocity unconstrained and ignores their error weights such that dynamics are represented by the first two components of (16). In this case the goal is to minimize a norm on the cart's acceleration while simultaneously minimizing the trajectory error in the pendulum angle relative to the origin (inverted equilibrium). The objective used to compare algorithm performance,

$$J_{pend} = \frac{1}{2}\int_{t_0}^{t_f} \|x(t) - x_d(t)\|_Q^2 + \|u(t)\|_R^2 \, dt\,, \qquad (19)$$

is applied in discretized form for SQP and iLQG results. This objective and weights $Q = Diag[1000,\, 10]$ and $R = [0.3]$ also provide the finite horizon cost for SQP and iLQG. All algorithms are constrained to provide controls $\ddot{x}_c \leq |25|\, \frac{m}{s^2}$.

Both SQP and iLQG require an a-priori choice of discretization and optimization horizon. For comparison, results are provided for different discretizations, dt, and optimization horizons, T.[11] Although SAC runs at 1 KHz, optimal control results are limited to dt$\geq$ 0.003 s, as SQP computations become infeasible and consume all computational resources below this.[12] Table II provides the time and number of optimization iterations required for each parameter combination.

dt		$T = 4$ s		$T = 5$ s		$T = 6$ s	
		min	iters	min	iters	min	iters
.01 s	SQP	13	1,234	22	1052	46	1,346
	iLQG	2	737	9	2,427	13	3,108
.005 s	SQP	169	2,465	32	201	105	251
	iLQG	5	908	56	8,052	5	622
.003 s	SQP	689	2,225	817	853	1,286	933
	iLQG	9	1,007	28	2,423	9	688

TABLE II: SQP versus iLQG for swing-up of the cart-pendulum under varying horizon, T, and discretization, dt. All solutions converge to the same optimizer except the gray results, which converged to low performance local minima.

The parameter combinations in Table II that do not correspond to gray data converged to the same (best case) optimal

[11] Horizons are based on the assumed time for pendulum inversion, and discretizations on assumed frequency requirements and linearization accuracy.

[12] All results were obtained on the same laptop with Intel® Core™ i7-4702HQ CPU @ 2.20GHz × 8 and 16GB RAM.

trajectory ($J_{pend} \approx 2,215$). Gray data indicate convergence to an alternate local minima with significantly worse cost. In all cases with $T \neq 4\,\text{s}$, SQP converges to local minima with costs $J_{pend} \approx 3,981 - 6,189$. While iLQG tends to be less sensitive to local minima, it converges to the worst local minima with $J_{pend} \approx 9,960$ for both finer discretizations when $T = 6\,\text{s}$.

Similarly, SAC control simulations included a variety of parameter combinations and moving horizons from $T = 0.15\,\text{s} - 3\,\text{s}$. These solutions yield costs[13] ranging from $J_{pend} = 2,215 - 2,660$, with the majority of solutions close or equal to $J_{pend} = 2,215$. The SAC solution depicted in Fig. 6 achieves the best case cost of $J_{pend} = 2,215$ from moving horizons of $T = 0.28\,\text{s}$, with the quadratic cost (12) and parameters $Q = 0$, $P_1 = Diag[500,\, 0]$, and $R = [0.3]$. Results show SAC develops closed-loop controls on-line that perform constrained inversion as well as the best solutions from offline optimal control. Also, local minima significantly affect SQP and iLQG, while SAC tends to be less sensitive.

Considering the simplicity of this nonlinear example, it is noteworthy that both optimal control algorithms require significant time to converge. While iLQG ranges from minutes to an hour, with a discretization $3\times$ as coarse as SAC, SQP requires $\approx$ 12 hours to compute the single, open-loop optimal trajectory in Fig. 6 utilizing 4 CPU cores. Our C++ implementation of SAC obtains a solution equivalent to the best results on-line with feedback at 1 KHz in less than $\frac{1}{2}$ s using 1 CPU core. [14] Computing optimal actions in closed-form, SAC achieves dramatic gains and avoids the iterative optimization process, which requires thousands of variables and constraints in SQP / iLQG.

Finally, we emphasize the closed-loop nature of SAC compared to SQP, which provides an open-loop trajectory, and iLQG, which yields an affine controller with both feedforward and feedback components. As the affine controller from iLQG is only valid in a local neighborhood of the optimal solution (SAC provides feedback on-line from arbitrary states), SQP or iLQG must be applied in moving horizon for feedback comparable to SAC. For improved speed, [67] recommends a moving horizon implementation using suboptimal solutions obtained after a fixed number (one) iteration. Though there are times when iterative routines approach a final solution well before convergence, this was not the case here. In this simple nonlinear example, SQP / iLQG trajectories only resembled the final solution a few iterations before convergence. Hence, implementing either algorithm in moving horizon is likely to result in a poor local solution (especially considering sensitivity to local minima).

3) Sensitivity to Initial Conditions: Using a prediction horizon $T = 1.2\,\text{s}$, SAC was applied to invert the same, reduced cart-pendulum system from a variety of initial conditions. Simulations used the quadratic tracking cost (12) and weight matrices from (19). A total of 20 initial conditions for $\theta(t)$, uniformly sampled over $[0, 2\,\pi)$, were paired with initial angular velocities at 37 points uniformly sampled over $[0, 4\,\pi]$.

To gauge performance, a $10\,\text{s}$ closed-loop trajectory was constructed from each of the 740 sampled initial conditions, and the state at the final time $x(10\,\text{s})$ measured. If the final state was within 0.001 rad of the inverted position and the absolute value of angular velocity was $< 0.001\,\frac{\text{rad}}{\text{s}}$, the trajectory was judged to have successfully converged to the inverted equilibrium. Tests confirmed the SAC algorithm was able to successfully invert the pendulum within $10\,\text{s}$ from all initial conditions. The average computation time was $\approx 1\,\text{s}$ for each $10\,\text{s}$ trajectory on the test laptop.

[13] In comparing SAC and SQP solutions, costs J_{pend} are computed from the SQP weight matrices for both cases.

[14] As the MATLAB SQP and iLQG implementations utilize compiled and parallelized libraries, it is unclear how to provide a side-by-side comparison to the timing results in Table II. To illustrate that SAC is still fast in slower, interpreted code, we also implemented SAC in Mathematica. Computations require $5 - 35\,\text{s}$ and are linear w.r.t. to horizon, T, and discretization, t_s.

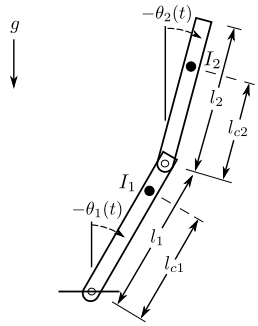


Fig. 7: Configuration of the acrobot and pendubot systems.

B. Pendubot and Acrobot

This section applies SAC for swing-up control of the pendubot [1], [48], [61] and the acrobot [59], [78], [79]. The pendubot is a two-link pendulum with an input torque that can be applied about the joint constraining the first (base) link. The acrobot is identical except the input torque acts about the second joint. The nonlinear dynamics and pendubot model parameters match those from simulations in [1] and experiments in [48]. The acrobot model parameters and dynamics are from simulations in [78] and in seminal work [59]. Figure 7 depicts the configuration variables and the model parameters are below. Each system's state vector is $x = (\theta_1,\, \dot{\theta}_1,\, \theta_2,\, \dot{\theta}_2)$ with the relevant joint torque control, $u = (\tau)$.

pendubot:	m_1 = 1.0367 kg	m_2 = 0.5549 kg
	l_1 = 0.1508 m	l_2 = 0.2667 m
	l_{c1} = 0.1206 m	l_{c2} = 0.1135 m
	I_1 = 0.0031 kg m^2	I_2 = 0.0035 kg m^2
acrobot:	m_1 = 1 kg	m_2 = 1 kg
	l_1 = 1 m	l_2 = 2 m
	l_{c1} = 0.5 m	l_{c2} = 1 m
	I_1 = 0.083 kg m^2	I_2 = 0.33 kg m^2

Due to their underactuated dynamics and many local minima, the pendubot and acrobot provide challenging test systems for control. As a popular approach, researchers often apply energy based methods for swing-up control and switch to LQR controllers for stabilization in the vicinity of the inverted equilibrium (see [1], [19], [36], [59], [61], [78], and [79]). We also use LQR controllers to stabilize the systems

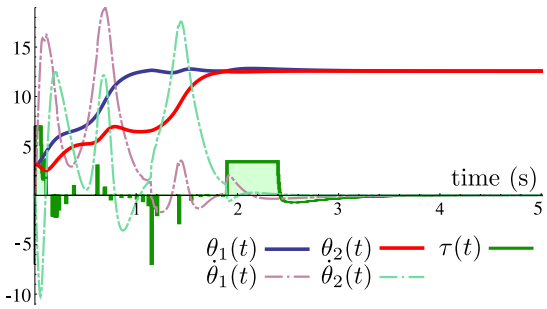


Fig. 8: SAC swings up the pendubot close enough for final stabilization by the LQR controller. The LQR controller takes effect at $t = 1.89$ s. The algorithm inverts the system using less peak control effort and in less time than existing methods from literature with the same parameters.

once near the inverted equilibrium. However, the results here show the SAC algorithm can swing-up both systems without relying on special energy optimizing methods. The algorithm utilizes the quadratic state error based cost functional (12), without modification.

While the pendubot simulations in [1] require control torques up to a magnitude of $15\,\mathrm{N\,m}$ for inversion, the experimental results in [48] perform inversion with motor torques restricted to $\pm 7\,\mathrm{N\,m}$. Hence, the pendubot inputs are constrained to $\tau \in [-7, 7]\,\mathrm{N\,m}$. The acrobot torques are constrained with $\tau \in [-15, 15]\,\mathrm{N\,m}$ to invert the system using less than the $20\,\mathrm{N\,m}$ required in [78].

Example simulations initialize each system at the downward, stable equilibrium and the desired position is the equilibrium with both links fully inverted. Results are based on a feedback sampling rate of 200 Hz for the pendubot with $Q = Diag[100,\, 0.0001,\, 200,\, 0.0001]$, $P_1 = 0$, and $R = [0.1]$ and 400 Hz for the acrobot with $Q = Diag[1,000,\, 0,\, 250,\, 0]$, $P_1 = Diag[100,\, 0,\, 100,\, 0]$, and $R = [0.1]$. The LQR controllers derived offline for final stabilization, $K_{lqr} = (-0.23,\, -1.74,\, -28.99,\, -3.86\,)$ and $K_{lqr} = (-142.73,\, -54.27,\, -95.23,\, -48.42\,)$, were calculated about the inverted equilibrium to stabilize the pendubot and acrobot systems, respectively. We selected $|\theta_{1,2}| \leq 0.05$ as the switching condition for pendubot stabilization.[15] the acrobot switched to stabilizing control once all its configuration variables were $\leq |0.25|$.

Figure 8 shows the pendubot trajectory (the acrobot and pendubot solutions are in video attachment). In both cases, SAC swings each system close enough for successful stabilization. With the same parameters, SAC inverts the pendubot system using the same peak effort as in experiments from [48] and less than half that from simulations in [1]. Also, SAC requires only $3\,\mathrm{s}$ to invert, while simulations in [1] needed $\approx 4\,\mathrm{s}$. Where the approach from [1] requires switching between separately derived controllers for pumping energy into, out of, and inverting the system before final stabilization, *SAC performs all these tasks without any change in parameters* and with the simple state tracking norm in (12). In the case of the acrobot, SAC inverts the system with the desired peak torque magnitude of $15\,\mathrm{N\,m}$ ($\frac{3}{4}$ the torque required in simulations from [78]). These closed-loop results were computed on-line and required only 1.23 and $4.7\,\mathrm{s}$ to compute $20\,\mathrm{s}$ trajectories for the pendubot and acrobot systems, respectively.

To invert the pendubot and acrobot in minimal time and under the tight input constraints, the two most important parameters for tuning are the desired change in cost due to each control actuation, α_d, and horizon length T. All examples specify α_d iteratively based on the current initial trajectory cost under the nominal (null) control as $\alpha_d = \gamma\, J_{1,init}$. From experimentation, $\gamma \in [-15, -1\,]$ tends to work best, but because of the speed of SAC computations, good parameter values can be found relatively quickly using sampling. These pendubot and acrobot results use $\gamma = -15$ and similar horizons of $T = 0.5\,\mathrm{s}$ and $T = 0.6\,\mathrm{s}$, respectively.

As mentioned earlier, optimal controllers typically use energy metrics for swing-up of the pendubot and acrobot, as simple state-tracking objectives yield local minima and convergence to undesirable solutions. It is noteworthy that SAC is able to invert both systems on-line and at high frequency considering optimal controllers (SQP/iLQG) generally fail under the same objective (12).

Part II: Extension to Hybrid Impulsive Systems

Part II of this paper extends SAC to systems with hybrid impulsive dynamics. These systems model a more general class of robotics problems in locomotion and manipulation, which involve contact and impacts. Such systems are challenging in optimal control and require specialized treatment and optimality conditions [35], [65], [66]. Because SAC plans infinitesimal actions individually, SAC does not need to optimize control curves over discontinuous segments of the trajectory.

IV. The Hybrid Mode Insertion Gradient

The SAC algorithm introduced in Section II is limited to differentiable nonlinear systems because the mode insertion gradient (4) is subject to Assump. 1. Derivations included in the following Section IV-A parallel the approach for continuous systems presented in [41] to develop a first-order approximation of the variation in state and cost due to perturbations in a nominal control generated by infinitesimal SAC actions. This section derives an equation for an adjoint variable similar to (5), but which applies to hybrid impulsive systems. The formula for the sensitivity of a cost functional (3) to infinitesimal control actions remains the same as the mode insertion gradient formula (4) assuming the adjoint variable is modified for hybrid impulsive systems. As a result (and a benefit of SAC), the SAC process described in Algorithm 1 remains unchanged for hybrid impulsive systems.

Section V demonstrates the hybrid system calculations on a $1D$ system and then illustrates the SAC approach in simulated on-line control of a bouncing ball. The section concludes with an example in which SAC controls a spring-loaded inverted pendulum (SLIP) model.

[15] More formally, a supervisory controller can switch between swing-up and stabilizing based on the stabilizing region of attraction [60], [62].

A. Definitions and Notation

The classes of hybrid systems considered here are similar to those in [65] and are defined such that:[16]

1) $\mathcal{Q}$ is a finite set of *locations*.
2) $\mathcal{M} = \{\mathcal{M}_q \subseteq \mathbb{R}^{n_q}\}_{q\in\mathcal{Q}}$ is a family of *state space* manifolds indexed by q.
3) $U = \{U_q \subset \mathbb{R}^{m_q}\}_{q\in\mathcal{Q}}$ is a family of *control spaces*.
4) $f = \{f_q \in \mathcal{C}(\mathcal{M}_q \times U_q, T\mathcal{M}_q)\}_{q\in\mathcal{Q}}$ is a family of maps to the tangent bundle, $T\mathcal{M}_q$. The maps $f_q(x,u) \in T_x\mathcal{M}_q$ are the *dynamics* at q.
5) $\mathcal{U} = \{\mathcal{U}_q \subseteq \mathcal{L}(\subset \mathbb{R}, U_q)\}_{q\in\mathcal{Q}}$ is a family of sets of admissible control mappings.
6) $\mathcal{I} = \{\mathcal{I}_q \subset \mathbb{R}^+\}_{q\in\mathcal{Q}}$ is a family of consecutive subintervals corresponding to the time spent at each location q.
7) The series of *guards*, $\Phi = \{\Phi_{q,q'} \in \mathcal{C}^1(\mathcal{M}_q, \mathbb{R}) : (q,q') \in \mathcal{Q}\}$, indicates transitions between locations q and q' when $\Phi_{q,q'}(x) = 0$. The state transitions according to a series of corresponding *reset maps*, $\Omega = \{\Omega_{q,q'} \in \mathcal{C}^1(\mathcal{M}_q, \mathcal{M}_{q'}) : (q,q') \in \mathcal{Q}\}$.

This section introduces new notation more appropriate for hybrid impulsive systems. For the sake of clarity, we avoid using numerical subscripts for the nominal control, u_1, from Section II. Instead, this section assumes a series of (possibly null) nominal controls, $\{u_{n,q} \in \mathcal{U}_q\}_{q\in\mathcal{Q}}$, exists for each location. With an initial location q_1, state $x(t_0) = x_{init} \in \mathcal{M}_{q_1}$, and the collection $\{f, \Phi, \Omega\}$, the location sequence, $q = (q_1, \ldots, q_r) : r \in \mathbb{N}$ and intervals, $\mathcal{I}$, exist and are defined based on the nominal state trajectory, $x_n(t) = x_{n,q_i}(t) : i \in \{1, \ldots, r\}, t \in \mathcal{I}_{q_i}$ from simulation of,

$$\dot{x}_{n,q_i} = f_{q_i}(x_{n,q_i}, u_{n,q_i}) : t \in \mathcal{I}_{q_i}\,. \tag{20}$$

Guards, Φ, indicate when a transition should occur (they specify the end of each interval $\mathcal{I}_{q_i}$) and the next location, q_{i+1}, in the sequence q (based on which guard becomes 0). Reset maps define the initial condition for evolution of the state according to (20) in location q_{i+1} as $\{x_{n,q_{i+1}}(t_i^+) = \Omega_{q_i,q_{i+1}}(x_{n,q_i}(t_i^-)) : t_i^- \triangleq \sup\mathcal{I}_{q_i}, t_i^+ \triangleq \inf\mathcal{I}_{q_{i+1}}\}$.

Infinitesimal actions in SAC are needle perturbations [50] relative to a nominal control. To model the effects of these needle perturbations in the nominal control, $u_n(t) = u_{n,q_i}(t) : t \in \mathcal{I}_{q_i}$, to first order, a perturbed control signal is defined,

$$u_w \triangleq \begin{cases} u_n & : t \notin [\tau - \epsilon a, \tau] \\ w & : t \in [\tau - \epsilon a, \tau] \end{cases},$$

for a (short) duration $\lambda = \epsilon a$. The magnitude of λ is specified as $\epsilon \in \mathbb{R}^+$ and the direction by an arbitrary positive scalar $a \in \mathbb{R}^+$. Because the perturbed system will eventually be evaluated for $\lambda \to 0^+$, assume the perturbation occurs in the arbitrary location q_i associated with the state $x_n(\tau)$ so that $[\tau - \epsilon a, \tau] \subseteq \mathcal{I}_{q_i}$. Figure 9 depicts the perturbed control and the corresponding perturbed (1D) state.

[16]We assume actions are not applied at switching times, exclude Zeno behavior, and allow only a single element of Φ to be active (zero) at a time to exclude simultaneous events and potentially indeterminate behavior. These (and continuity) assumptions guarantee a local neighborhood exists such that perturbed system trajectories evolve through the same location sequence (as required in [65]).

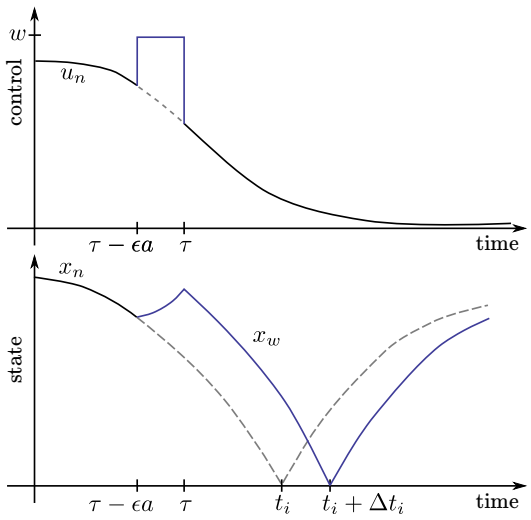


Fig. 9: A perturbed control (top) and the corresponding (hypothetical) state variation (bottom) for a hybrid system. The nominal system switches locations at time t_i and the perturbed system switches at time $t_i + \Delta t$. Taken in the limit as $\epsilon a \to 0^+$, the control perturbation is a needle perturbation that is equivalent to an infinitesimal action in SAC.

Section IV-B develops a first-order perturbed state model,[17]

$$x_w(t,\epsilon) \triangleq x_n(t) + \epsilon\Psi(t) + o(\epsilon)\,. \tag{21}$$

To solve for the direction of state variations, Ψ, the section follows the continuous systems approach in [41] and uses first-order Taylor expansions to approximate $\Psi(\tau)$.

B. Initial Condition for State Variations

Expanding x_n about $t = \tau$, and (21) about $t = \tau - \epsilon a$ yields

$$x_n(\tau - \epsilon a) = x_n(\tau) - f_{q_i}(x_n(\tau), u_n(\tau))\epsilon a + o(\epsilon) \tag{22}$$

and

$$x_w(\tau,\epsilon) = x_n(\tau - \epsilon a) + f_{q_i}(x_n(\tau - \epsilon a), w)\epsilon a + o(\epsilon)\,. \tag{23}$$

Similarly expanding $f_{q_i}(x_n(\tau - \epsilon a), w)$ around $x_n(\tau)$,

$$\begin{aligned} f_{q_i}(x_n(\tau - \epsilon a), w) = {} & f_{q_i}(x_n(\tau), w) \\ & + D_x f_{q_i}(x_n(\tau), w)\big[x_n(\tau - \epsilon a) \\ & - x_n(\tau)\big] + o(x_n(\tau - \epsilon a) - x_n(\tau))\,, \end{aligned}$$

and applying $x_n(\tau - \epsilon a) - x_n(\tau) = -f_{q_i}(x_n(\tau), u_n(\tau))\epsilon a + o(\epsilon)$ from (22), one can simplify (23) so that,

$$x_w(\tau,\epsilon) = x_n(\tau - \epsilon a) + f_{q_i}(x_n(\tau), w)\epsilon a + o(\epsilon)\,. \tag{24}$$

Plugging (22) into (24) results in a first-order approximation of the perturbation at time $t = \tau$,

$$x_w(\tau,\epsilon) = x_n(\tau) + \Big(f_{q_i}(x_n(\tau), w) - f_{q_i}(x_n(\tau), u_n(\tau))\Big)\epsilon a + o(\epsilon)\,. \tag{25}$$

[17]The litte-o notation, $o(\epsilon)$, indicates terms that are higher than first order in ϵ. These terms go to zero faster than first-order terms in (21) as $\epsilon \to 0$.

Based on (21), the formula (25) indicates the initial direction for state variations,

$$\Psi(\tau) = \Big(f_{q_i}(x_n(\tau), w) - f_{q_i}(x_n(\tau), u_n(\tau)) \Big) a \,. \tag{26}$$

C. Propagation of Variations Within a Location

Assuming the variation occurs at $t = \tau$ in location $q_i \in \mathcal{Q}$, the varied state propagates according to

$$x_w(t, \epsilon) = x_w(\tau, \epsilon) + \int_\tau^t f_{q_i}(x_w(s, \epsilon), u_n(s)) ds : t \in \mathcal{I}_{q_i} \,. \tag{27}$$

By (21), $\Psi(t) = D_\epsilon x_w(t, \epsilon)\big|_{\epsilon \to 0}$, hence differentiating (27) develops the *variational equation* [41], [50],

$$\begin{aligned} D_\epsilon x_w(t, 0) &= D_\epsilon x_w(\tau, 0) \\ &\quad + \int_\tau^t D_x f_{q_i}(x_w(s, 0), u_n(s)) D_\epsilon x_w(s, 0) ds \\ \Psi(t) &= \Psi(\tau) + \int_\tau^t D_x f_{q_i}(x_n(s), u_n(s)) \Psi(s) ds \\ &= \Psi(\tau) + \int_\tau^t A_{q_i}(s) \Psi(s) ds \,. \end{aligned} \tag{28}$$

The term $A_{q_i}(t) \triangleq D_x f_{q_i}(x_n(t), u_n(t)) : t \in \mathcal{I}_{q_i}$ is the linearization about the (known) nominal state trajectory at q_i. Based on the initial condition (26), (28) is the solution to,

$$\dot{\Psi} = A_{q_i} \Psi : t \in \mathcal{I}_{q_i} \,, \tag{29}$$

which governs the flow of the first-order state variation at q_i.

D. Propagation of Variations Through Locations

Assuming the control perturbation occurs at $t = \tau$ in location $q_i \in \mathcal{Q}$, the expression,

$$\begin{aligned} x_w(t, \epsilon) &= x_w(t_i + \Delta t_i^+, \epsilon) + \int_{t_i + \Delta t_i^+}^t f_{q_{i+1}}(x_w(s, \epsilon), u_n(s)) ds \\ &= \Omega_{q_i, q_{i+1}}(x_w(t_i + \Delta t_i^-, \epsilon)) \\ &\quad + \int_{t_i + \Delta t_i^+}^t f_{q_{i+1}}(x_w(s, \epsilon), u_n(s)) ds \\ &= \Omega_{q_i, q_{i+1}} \left(x_w(t_i, \epsilon) + \int_{t_i}^{t_i + \Delta t_i^-} f_{q_i}(x_w(s, \epsilon), u_n(s)) ds \right) \\ &\quad + \int_{t_i + \Delta t_i^+}^t f_{q_{i+1}}(x_w(s, \epsilon), u_n(s)) ds \quad : t \in \mathcal{I}_{q+1} \,, \end{aligned} \tag{30}$$

propagates the varied system to location q_{i+1}. Note t_i is the time at which the nominal (unperturbed) state, x_n, transitions between locations q_i and q_{i+1}, $\Delta t_i \triangleq \Delta t_i(\epsilon)$ is the change in transition time due to the state variations, and a "$-$" or "$+$" superscript (as in $t_i + \Delta t_i^+$) indicates the time just before or after a transition. Figure 9 shows how the control perturbation causes an update in transition time from t_i to $t_i + \Delta t_i$ for the perturbed state. The figure depicts a scenario where the reset map, $\Omega_{q_i, q_{i+1}}$, causes the state's velocity to reflect (as in a collision) when $\Phi_{q_i, q_{i+1}}(x) \triangleq x : x \in \mathcal{M}_{q_i}$ becomes 0 at $x_w(t_i + \Delta t_i) = 0$.

As before, differentiating (30) as $\epsilon \to 0$ provides the first-order variational equation for Ψ at q_{i+1} due to a control perturbation at q_i,

$$\begin{aligned} D_\epsilon x_w(t, 0) &= D_x \Omega_{q_i, q_{i+1}}(x_w(t_i^-, 0)) \frac{d x_w(t_i + \Delta t_i^-, \epsilon)}{d\epsilon}\bigg|_{\epsilon \to 0} \\ &\quad + \int_{t_i + \Delta t_i^+}^t D_x f_{q_{i+1}}(x_w(s, 0), u_n(s)) D_\epsilon x_w(s, 0) ds \\ &\quad - \frac{d \Delta t_i^+}{d\epsilon}\bigg|_{\epsilon \to 0} f_{q_{i+1}}(x_n(t_i^+), u_n(t_i^+)) \\ \Psi(t) &= D_x \Omega_{q_i, q_{i+1}}(x_n(t_i^-)) \Big[\Psi(t_i^-) \\ &\quad + \frac{d \Delta t_i}{d\epsilon}\bigg|_{\epsilon \to 0} f_{q_i}(x_n(t_i^-), u_n(t_i^-)) \Big] \\ &\quad - \frac{d \Delta t_i}{d\epsilon}\bigg|_{\epsilon \to 0} f_{q_{i+1}}(x_n(t_i^+), u_n(t_i^+)) \\ &\quad + \int_{t_i^+}^t A_{q_{i+1}}(s) \Psi(s) ds \quad : t \in \mathcal{I}_{q_{i+1}} \,. \end{aligned} \tag{31}$$

This variational equation requires $\frac{d \Delta t_i}{d\epsilon}\big|_{\epsilon \to 0}$. The term is obtained by locally enforcing the guard equation,

$$\Phi_{q_i, q_{i+1}}(x_w(t_i + \Delta t_i^-, \epsilon)) = 0 \,, \tag{32}$$

using the first-order Taylor expansion of (32) around $\epsilon \to 0$,

$$\begin{aligned} 0 &= \Phi_{q_i, q_{i+1}}(x_n(t_i^-)) \\ &\quad + D_x \Phi_{q_i, q_{i+1}}(x_n(t_i^-)) \Big[f_{q_i}(x_n(t_i^-), u_n(t_i^-)) \frac{d \Delta t_i}{d\epsilon}\bigg|_{\epsilon \to 0} \\ &\quad + \Psi(t_i^-) \Big] + o(\epsilon) \,. \end{aligned}$$

Applying $\Phi_{q_i, q_{i+1}}(x_n(t_i^-)) = 0$ in the expansion yields,

$$\frac{d \Delta t_i}{d\epsilon}\bigg|_{\epsilon \to 0} = - \frac{D_x \Phi_{q_i, q_{i+1}}(x_n(t_i^-)) \Psi(t_i^-)}{D_x \Phi_{q_i, q_{i+1}}(x_n(t_i^-)) f_{q_i}(x_n(t_i^-), u_n(t_i^-))} \,. \tag{33}$$

Finally, one can define a new reset term,

$$\begin{aligned} \Pi_{q_i, q_{i+1}} &\triangleq D_x \Omega_{q_i, q_{i+1}}(x_n(t_i^-)) \Big[I - f_{q_i}(x_n(t_i^-), u_n(t_i^-)) \\ &\quad \frac{D_x \Phi_{q_i, q_{i+1}}(x_n(t_i^-))}{D_x \Phi_{q_i, q_{i+1}}(x_n(t_i^-)) f_{q_i}(x_n(t_i^-), u_n(t_i^-))} \Big] \\ &\quad + f_{q_{i+1}}(x_n(t_i^+), u_n(t_i^+)) \\ &\quad \frac{D_x \Phi_{q_i, q_{i+1}}(x_n(t_i^-))}{D_x \Phi_{q_i, q_{i+1}}(x_n(t_i^-)) f_{q_i}(x_n(t_i^-), u_n(t_i^-))} \,, \end{aligned} \tag{34}$$

such that plugging (28) and (33) into (31) reveals

$$\begin{aligned} \Psi(t) &= \Pi_{q_i, q_{i+1}} \left[\Psi(\tau) + \int_\tau^{t_i^-} A_{q_i}(s) \Psi(s) ds \right] \\ &\quad + \int_{t_i^+}^t A_{q_{i+1}}(s) \Psi(s) ds \\ &= \Pi_{q_i, q_{i+1}} \Psi(t_i^-) + \int_{t_i^+}^t A_{q_{i+1}}(s) \Psi(s) ds \quad : t \in \mathcal{I}_{q_{i+1}} \,. \end{aligned} \tag{35}$$

Just as $\Omega_{q_i, q_{i+1}}$ resets the state in (30), $\Pi_{q_i, q_{i+1}}$ provides a (linear) reset map for variations that transitions them between locations in (35).

Rather than calculate Ψ from (35), one can compute variations from a series of differential equations and resets. When the control perturbation occurs in location q_i, Ψ flows to q_{i+1} based on,

$$\Psi \triangleq \begin{cases} \Big(f_{q_i}(x_n(\tau), w) - f_{q_i}(x_n(\tau), u_n(\tau))\Big) a & : t = \tau \in \mathcal{I}_{q_i} \\ \dot{\Psi} = A_{q_i} \Psi & : t \in (\tau, t_i^-] \\ \Psi(t_i^+) = \Pi_{q_i, q_{i+1}} \Psi(t_i^-) & : t = t_i^+ \\ \dot{\Psi} = A_{q_{i+1}} \Psi & : t \in (t_i^+, t_{i+1}^-] \end{cases} . \tag{36}$$

Note that if the nominal trajectory evolves through additional locations, each transition requires reset of Ψ at transition times according to (34). Variations continue according to the dynamics linearized about the nominal trajectory. Repeating computations in rows $2-4$ of (36) between consecutive locations, variations can be propagated to $t = t_f$.

E. Sensitivity to Variations

Assume a series of *incremental costs*, $\{l_{q_i} \in \mathcal{C}^1(\mathcal{M}_{q_i} \times U_{q_i}, \mathbb{R})\}_{q_i \in \mathcal{Q}}$, such that $l = l_{q_i} : t \in \mathcal{I}_{q_i}$. Given $(x_n, q, \mathcal{I})$ resulting from nominal control, u_n, the objective,

$$J = \int_{t_0}^{t_f} l(x(t), u(t)) dt \,, \tag{37}$$

measures performance of the hybrid trajectory. Note that by appending the incremental costs, l_{q_i}, to the dynamics vectors, f_{q_i}, in each location, the hybrid system is translated to Mayer form with (37) as an appended state. Objects with a bar refer to appended versions of hybrid system such that $\bar{f}_{q_i} = [\, l_{q_i},\, f_{q_i}^T \,]^T$, $\bar{x} = [\, J,\, x^T \,]^T$, and

$$\bar{A}_{q_i} = \begin{pmatrix} 0 & D_x l_{q_i} \\ 0 & A_{q_i} \end{pmatrix} \Big|_{(x_n, u_n)} .$$

In Mayer form, it is straightforward to compute the first-order variation, $\delta J \triangleq \epsilon \nu$, in system performance (37) due to a needle perturbation in the control at arbitrary (given) time $\tau \in [t_0, t_f]$. One need only propagate appended state variations using the methods just introduced. The first component of $\bar{\Psi}(t_f)$ provides the direction, $\nu(t_f)$, of the variation in (37) and ϵ is its magnitude. However, computing the variation direction, $\nu(t_f)$, due to a control perturbation at a different time $\tau' \neq \tau$, requires re-simulation of $\bar{\Psi}$ from the new initial condition at $\bar{\Psi}(\tau')$. Hence, the process is computationally intensive.

To compute the first-order variation direction, $\nu(t_f)$, resulting from a control needle variation at an arbitrary, unknown time, $\tau \in [t_0, t_f]$, we derive an *adjoint* system, $\bar{\rho}$,[18] to the variational system $\bar{\Psi}$.[19] The two systems are adjoint [41] if,

$$\frac{d}{dt}(\bar{\rho} \cdot \bar{\Psi}) = 0 = \dot{\bar{\rho}} \cdot \bar{\Psi} + \bar{\rho} \cdot \dot{\bar{\Psi}} \,. \tag{38}$$

That is, $\bar{\rho} \cdot \bar{\Psi}$ is constant, a property that will be enforced through differentiation in deriving the adjoint system. With a terminal condition, $\bar{\rho}(t_f)$, in which the first element of $\bar{\rho}(t_f)$ is 1 and the remaining elements are 0, the inner product $\bar{\rho}(t_f) \cdot \bar{\Psi}(t_f) = \nu(t_f)$. Because of (38), the inner product is constant and equal to $\nu(t_f)$ at times subsequent to the control perturbation, $\bar{\rho}(t) \cdot \bar{\Psi}(t) = \nu(t_f)\ \forall t \in [\tau, t_f]$, and $\bar{\rho}$ evaluated at any time is the sensitivity of (37) to the state variation at that time. These properties facilitate analysis of control perturbations at different times $\tau \in [t_0, t_f]$.

Assuming the system is at the (arbitrary) location $q \in \mathcal{Q}$ at the perturbation time $t = \tau$, the inner product in (38) yields

$$\begin{aligned} \bar{\rho}(\tau) \cdot \bar{\Psi}(\tau) &= \bar{\rho}(\tau) \cdot \Big(\bar{f}_q(x_n(\tau), w) - \bar{f}_q(x_n(\tau), u_n(\tau))\Big) a \\ &= \nu(t_f) \,. \end{aligned} \tag{39}$$

The initial time, τ, of the control perturbation is arbitrary. Evaluating (39) at any time $\tau \in [t_0, t_f]$ provides the sensitivity of (37) to the duration of a control needle perturbation applied at that time (i.e., it equals $\nu(t_f)$), similar to the mode insertion gradient (4). Considering two possible times $\tau < \tau'$ when needle variations may be applied, $\nu(t_f)$ would require separate simulations of $\bar{\psi}$ from $[\tau, t_f]$ and $[\tau', t_f]$.[20] In contrast, a single simulation of $\bar{\rho}$ from $[t_f, \tau]$ evaluated at τ and τ' in (39) provides the performance sensitivity, $\nu(t_f)$, for each case.

F. Relation to the Mode Insertion Gradient

In hybrid systems literature [12], [16], [22], [73], [72], the mode insertion gradient is derived for systems with incremental costs, l, which do not depend on the control (see (3)), and the state dimension is fixed, so $f_{q'}$ and f_q are of the same dimension $(q, q') \in \mathcal{Q}$. Under these assumptions, the mode insertion gradient provides the sensitivity of J to insertion of a different dynamic mode (i.e., switching from f_q to the dynamics $f_{q'}$ of an alternate location q' for a short duration around $\lambda \to 0^+$). In the case of SAC, the alternate dynamic modes differ only in control (i.e., $f_{q'} \triangleq f_q(x_n(\tau), w)$ and $f_q \triangleq f_q(x_n(\tau), u_n(\tau))$) and so result in the form of the mode insertion gradient in (4) for smooth systems. The expression (39) provides SAC the same information (the sensitivity of a trajectory cost, J, to applying an infinitesimal action at some time) as the form of the mode insertion gradient in (4) but applies to hybrid impulsive systems with resets.[21]

Using the methods presented, it is straightforward to modify the initial condition of the variational equation (26) to accommodate an arbitrary dynamic mode insertion, $f_{q'}$, rather an a control perturbation. Note the formulas for the variational flow (36) and its corresponding adjoint equation, which will be derived in the subsequent subsection, would remain unchanged. In this case, (39) becomes the more general form of the mode insertion gradient from hybrid systems literature (as it considers more than just control perturbations), but applies to broader classes of hybrid impulsive systems with resets.

[18] The adjoint belongs to the tangent bundle, $\bar{\rho} \in T^* \mathcal{M}_{q_i}$, such that $\bar{\rho}(t) : T_x \mathcal{M}_{q_i} \mapsto \mathbb{R},\ \forall t \in \mathcal{I}_{q_i}, \forall q_i \in q$.

[19] See [41] for a similar derivation of an adjoint in the context of continuous variations.

[20] One may also apply linear transformations to the variational system simulated from the perturbation at τ based on superposition of the initial condition at τ'. Variational reset maps would require similar transformation.

[21] As one would hope, (39) is equivalent to the mode insertion gradient formula in (4) when w corresponds to an optimal infinitesimal action, the incremental costs, l, do not depend on the control (see (3)), and the system is not hybrid (locations q_i and q_{i+1} are the same).

Hence, the derivations and hybrid adjoint and mode insertion gradient calculations (39) introduced in this section can enable mode scheduling algorithms like those in [12], [22], [73], [72] for these larger classes of hybrid and impulsive systems.

G. Adjoint Simulation

To compute the sensitivity of a cost functional to control perturbations using (39), we use the adjoint, $\bar{\rho}$, and maintain its interpretation as the cost sensitivity to state variations, as in $\bar{\rho}(t) \cdot \bar{\Psi}(t) = \nu(t_f)\ \forall t \in [\tau, t_f]$. First, assume the system is in location q_{i+1} at final time $t = t_f$ and that the needle perturbation in the control occurred at $t = \tau$ in the same location. The state variations will flow forward on $[\tau, t_f]$ with (26) (replacing q_i with q_{i+1} and the dynamics with appended versions) as an initial condition under dynamics $\dot{\bar{\Psi}} = \bar{A}_{q_{i+1}} \bar{\Psi}$. In this location, $\dot{\bar{\rho}} = -\bar{A}^T_{q_{i+1}} \bar{\rho}$ satisfies (38) and so is adjoint to $\bar{\Psi}$. With its dynamics defined, $\bar{\rho}$ is obtained by simulation from an appropriate initial / terminal condition. As described in Section IV-E, choosing $\bar{\rho}(t_f) = [1,\, 0,\, \ldots,\, 0]^T \in \mathbb{R}^{n_{q_{i+1}}}$ as the terminal condition maintains the desired relationship, $\bar{\rho}(t) \cdot \bar{\Psi}(t) = \nu(t_f)\ \forall t \in [\tau, t_f]$.

Now consider the case where the system is in location q_{i+1} at the final time, $t = t_f$, and that the needle perturbation in the control occurred at $t = \tau$ in location q_i. The variational system evolves according to (36). The adjoint system follows the same differential equation, $\dot{\bar{\rho}} = -\bar{A}^T_{q_{i+1}} \bar{\rho}$, and terminal condition as in the previous case. The adjoint flows backwards in time until the variational equation jumps at t_i. The corresponding adjoint reset map is derived by enforcing (38) at t_i. As such, the inner product of the adjoint and variational equations must be constant from $t \in [t_i^-, t_i^+]$,

$$\frac{d(\bar{\rho}(t_i) \cdot \bar{\Psi}(t_i))}{dt} = 0 = \frac{\bar{\rho}(t_i^+) \cdot \bar{\Psi}(t_i^+) - \bar{\rho}(t_i^-) \cdot \bar{\Psi}(t_i^-)}{t_i^+ - t_i^-}$$

$$0 = \bar{\rho}(t_i^+) \cdot \bar{\Pi}_{q_i, q_{i+1}} \bar{\Psi}(t_i^-) - \rho(t_i^-) \cdot \bar{\Psi}(t_i^-)$$

$$\bar{\rho}(t_i^-) = \bar{\Pi}^T_{q_i, q_{i+1}} \bar{\rho}(t_i^+)\,.$$

The final expression provides the reset map required to flow $\bar{\rho}$ backwards from t_i^+ in location q_{i+1} to t_i^- in location q_i. Once in location q_i, the flow of the adjoint evolves according to the dynamics $\dot{\bar{\rho}} = -\bar{A}^T_{q_i} \bar{\rho}$ to remain adjoint to $\bar{\Psi}$ until $t = \tau \in \mathcal{I}_{q_i}$. Thus the adjoint satisfies

$$\bar{\rho} \triangleq \begin{cases} [1,\, 0,\, \ldots,\, 0]^T & : t = t_f \\ \dot{\bar{\rho}} = -\bar{A}^T_{q_{i+1}} \bar{\rho} & : t \in (t_i^+, t_f) \\ \bar{\rho}(t_i^-) = \bar{\Pi}^T_{q_i, q_{i+1}} \bar{\rho}(t_i^+) & : t = t_i^- \\ \dot{\bar{\rho}} = -\bar{A}^T_{q_i} \bar{\rho} & : t \in [\tau, t_i^-) \end{cases} . \tag{40}$$

As for the variational equation, one may propagate $\bar{\rho}$ between arbitrary numbers of consecutive modes by repeating the reset and continuous flow steps.

H. A Terminal Objective

The previous subsection shows that deriving the adjoint equation based on the terminal condition $\bar{\rho}(t_f) = [1,\, 0,\, \ldots,\, 0]^T$ results in a constant inner product, $\bar{\rho}(t) \cdot \bar{\Psi}(t) = \nu(t_f)\ \forall t \in [\tau, t_f]$. It is this choice that allows interpretation of (39) as the sensitivity of the performance objective (37) to a needle variation in the control that may occur at any time $\tau \in [t_0, t_f]$. Similarly, $\bar{\rho}(\tau)$ is the sensitivity of the objective to state perturbations for $\tau \in [t_0, t_f]$. This subsection derives a terminal condition that maintains $\bar{\rho}(t) \cdot \bar{\Psi}(t) = \nu(t_f)$ and the meaning of $\bar{\rho}$ and (39) when (37) includes terminal cost mapping, m, such that $\{m_{q_i} \in \mathcal{C}^1(\mathcal{M}_{q_i}, \mathbb{R})\}_{q_i \in \mathcal{Q}}$, $m = m_{q_i} : t \in \mathcal{I}_{q_i}$, and

$$J = \int_{t_0}^{t_f} l(x(t), u(t))dt + m(x(t_f))\,. \tag{41}$$

Again, by setting the first element of $\bar{\rho}(t_f)$ to 1, the inner product $\bar{\rho}(t_f) \cdot \bar{\Psi}(t_f)$ includes the direction of variation in the incremental cost component, $\int_{t_0}^{t_f} l(x(t), u(t))dt$, when the state is in Mayer form. By adding the direction of variation in $m(x(t_f))$, one obtains the new variation direction $\nu(t_f)$ for (41). The derivative of $m(x(t_f))$ as $\epsilon \to 0$,

$$D_\epsilon m(x_w(t_f, 0)) = D_x m(x(t_f)) \Psi(t_f)\,, \tag{42}$$

provides the first-order direction of variation in the terminal cost. Note (42) is an inner product of the (known) gradient $\nabla m(x(t_f))$ and the direction of state variation at the final time $\Psi(t_f)$. Hence, the variation direction for the performance objective is provided as $\nu(t_f) = [\,1,\, D_x m(x(t_f))\,]\, \bar{\Psi}(t_f)$ when J includes a terminal cost. With the new terminal condition,

$$\bar{\rho}(t_f) = [\,1,\, D_x m(x(t_f))\,]^T, \tag{43}$$

the adjoint relationship ensures $\bar{\rho}(t) \cdot \bar{\Psi}(t)$ and (39) are equal to $\nu(t_f)\ \forall t \in [\tau, t_f]$ and $\forall \tau \in [t_0, t_f]$.

V. Hybrid Control Examples

This section presents three illustrative examples using the hybrid methods just described. Section V-A demonstrates calculation of the variational, adjoint, and *hybrid mode insertion gradient* (39) equations for a 1D example. Section V-B uses the hybrid version of SAC (based on the adjoint in (40)) to control a bouncing ball through impacts and toward a goal state. Lastly, Section V-C applies SAC to control a the hybrid spring-loaded inverted pendulum model up a flight of stairs.

A. Variations, Adjoint, and Control Sensitivity for a 1D Bouncing Mass

This section computes variational and adjoint equations for a simple point mass system with impacts. The point mass is released from a height of $z_0 = 1\,\text{m}$ with no initial velocity. It falls under gravity until it impacts with a flat surface (guard) at $z = 0\,\text{m}$. The dynamics before and after impact (locations q_1 and q_2, respectively) are the same, corresponding to a point mass in gravity. However, a reset map reflects velocity, $\dot{z}$, when the guard becomes 0 at impact. The simulation parameters follow.

System Parameters:

$x = (\, z,\, \dot{z}\,)$	$f_{q_1}(x, u) = (\, \dot{z},\, -g + u\,)$
$g = 9.81\ \frac{\text{m}}{s^2}$	$f_{q_2}(x, u) = f_{q_1}(x, u)$
$J = \int_{t_0}^{t_f} x^T Q\, x\, dt$	$Q = Diag[\,200,\, 0.01\,]$
$\Omega_{q_1, q_2}(x) = Diag[\,1,\, -1\,]\, x$	$\Phi_{q_1, q_2}(x) = z$

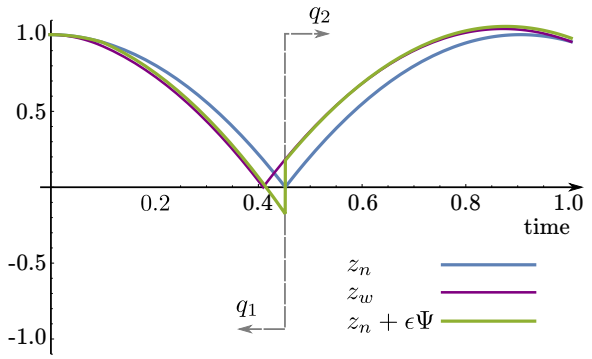


Fig. 10: The height $z(t)$ of a mass dropped from 1 m. The mass follows the nominal trajectory, z_n, and bounces at impact due to an elastic collision. The reset map reflects its velocity in transitioning from q_1 to q_2. The purple curve is the varied trajectory simulated from the hybrid impulsive dynamics with a needle variation at $\tau = 0.1\,\text{s}$ of duration $\lambda = 0.1\,\text{s}$ ($a = 1$, $\epsilon = 0.1\,\text{s}$), in the nominal control. The variation accelerates the mass in the z direction at $w = -5\,\frac{\text{m}}{\text{s}^2}$. The green curve is the approximated trajectory based on the first-order model.

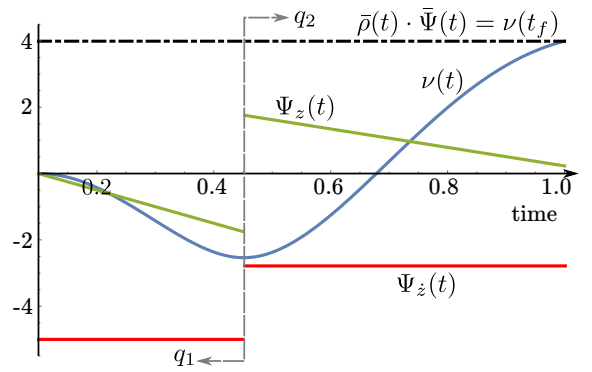


Fig. 11: The direction of state (and cost) variations, $\bar{\Psi} = (\,\nu,\, \Psi_z,\, \Psi_{\dot{z}}\,)$, resulting from a needle variation at $\tau = 0.1\,\text{s}$ of duration $\lambda = 0.1\,\text{s}$ ($a = 1$, $\epsilon = 0.1\,\text{s}$) in the nominal control. The control variation accelerates the falling mass in the z direction at $w = -5\,\frac{\text{m}}{\text{s}^2}$. At all times subsequent the control variation, $\forall t \in [\tau, t_f]$, $\bar{\rho}(t) \cdot \bar{\Psi}(t)$ is equal to the direction of variation in the cost propagated to the final time, $\nu(t_f)$. The state variations in z and $\dot{z}$ are discontinuous at the transition from q_1 to q_2, while the cost variation is continuous.

Control Perturbation:

$u_n = 0\,\frac{\text{m}}{\text{s}^2}$	$\tau = 0.1\,\text{s}$
$a = 1$	$\epsilon = 0.1\,\text{s}$
$w = -5\,\frac{\text{m}}{\text{s}^2}$	$\lambda = 0.1\,\text{s}$

Figure 10 shows the system's nominal trajectory (blue curve) and the varied trajectory resulting from a simulated needle variation in control. The varied trajectory is computed from both the first-order variational model (green curve) and the true, nonlinear hybrid impulsive dynamics (purple curve). The variation directions resulting from (36) are in Fig. 11. As Fig. 11 shows, the state variations are discontinuous at impact, while the direction of the cost variation, $\nu(t)$, is continuous

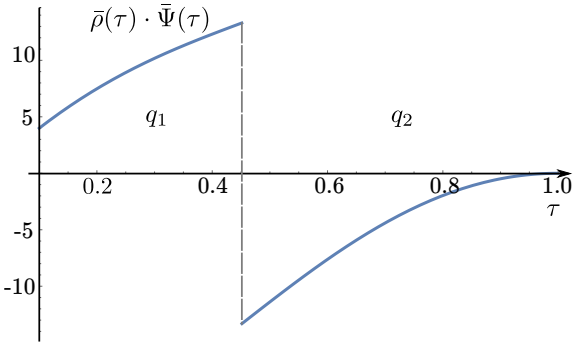


Fig. 12: The value of $\bar{\rho}(\tau) \cdot \bar{\Psi}(\tau)$ (according to (39)) versus τ. The term indicates the sensitivity of the performance objective to the control perturbation, $w = -5\,\frac{\text{m}}{\text{s}^2}$, if that perturbation were to occur at different points $\tau \in [t_0, t_f]$. Before impact, (39) indicates a short control perturbation will increase cost (41). After impact, a short control perturbation will lower cost.

over time. The dashed black line in Fig. 11 confirms the inner product, $\bar{\rho} \cdot \bar{\Psi}$, is constant and equal to the direction of the cost variation, $\nu(t_f)$, for all time subsequent the control perturbation, $\forall t \in [\tau, t_f]$. Figure 12 shows how this inner product (the value of (39)) would change if the control perturbation were applied at different times, $\tau \in [t_0, t_f]$.

Results:

$\bar{\rho}(\tau) \cdot \bar{\Psi}(\tau)\big|_{\tau=0.1} = 4$

$\nu(t_f) = [\,1,\, 0,\, 0\,]^T \cdot \bar{\Psi}(t_f) = 4$

$\Delta t_i \approx \epsilon \times \frac{d\Delta t_i}{d\epsilon}\big|_{\epsilon \to 0} = -0.04\,\text{s}$

$$\Pi_{q_1,q_2} = \begin{pmatrix} -1 & 0 \\ \frac{-2g+2u}{\dot{z}} & -1 \end{pmatrix}$$

As asserted earlier, the approximation of the change in cost (41) from (39) agrees with the first-order approximation of the change in cost from simulation of $\bar{\Psi}(t_f)$. The first-order variational model, $z_n + \epsilon\Psi$, in Fig. 10 closely approximates the true perturbed trajectory, z_w, simulated from the perturbed control and the nonlinear dynamics. Additionally, (33) estimates the impact time of the varied system as $t = 0.41\,\text{s}$, which is near the updated impact time of z_w in Fig. 10. Figure 12 shows that (39) correctly indicates it will be helpful (reduce trajectory cost according to (41)) to apply the control perturbation (push the mass toward the ground) after impact, when the ball is moving away from the ground. Similarly, the figure suggests it will be detrimental to apply the control perturbation before impact because it would result in a net gain (positive change) in trajectory cost according to the first-order model.

Finally, note that the reset map, Π_{q_1,q_2}, is only defined for velocities $\dot{z}$ that are non-zero. As is typical for hybrid systems, these methods require that some component of the system's velocity vector lie in the direction of the switching surface so as to preclude grazing impacts. The requirement ensures both (33) and (34) are well defined with $D_x\Phi_{q,q'}(x_n(t_i^-))f_{q_i}(x_n(t_i^-), u_n(t_i^-)) \neq 0\ \forall (q, q') \in \mathcal{Q}$.

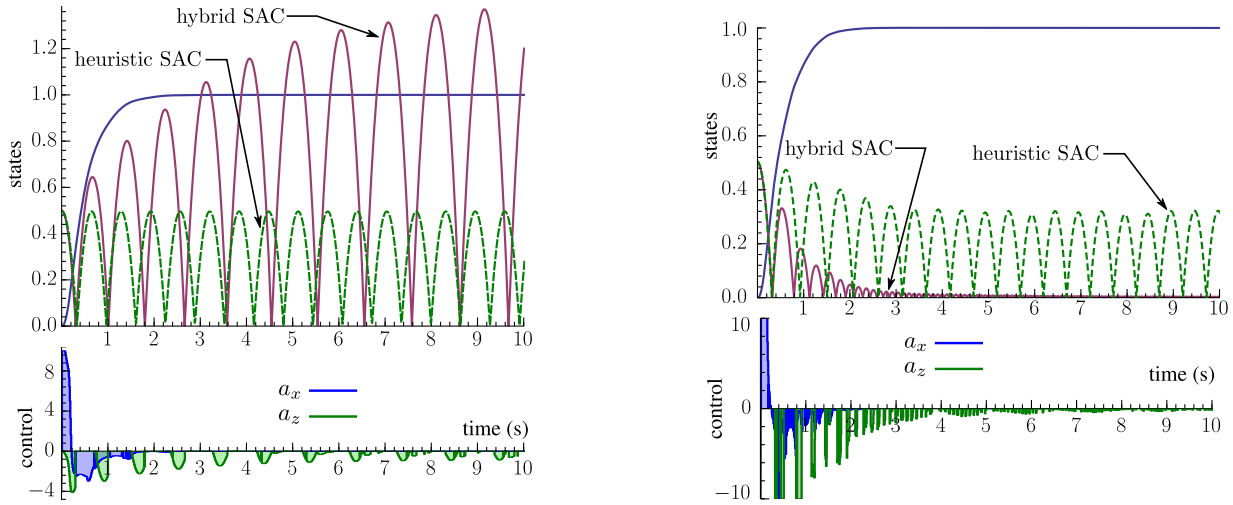


Fig. 13: SAC accelerates a ball 1 m to the right and either up (Fig. 13a) or down (Fig. 13b). In both cases x_b (the blue state curve) reaches the desired point 1 m away. In Fig. 13a, control constraints prohibit the ball from accelerating against gravity, and so it cannot come to rest at the desired height. Instead, SAC accelerates the ball into the floor to rebound, increasing its height, z_b (purple state curve), to maximize the time spent around the desired height of 1 m. If the smooth version of SAC is applied as a heuristic (without the hybrid modifications), SAC drives the ball to the desired horizontal position but will not thrust in the a_z direction. Hence, the ball will continuously bounce at the initial height. Similarly, in Fig. 13b, the hybrid version of SAC successfully reduces energy from the (conservative) system by accelerating the ball into the floor when its momentum is away from the floor. Though it gets indistinguishably close, the ball cannot come to rest on the ground or it would result in infinite switching. If the smooth version of SAC is applied as a heuristic, SAC will drive the ball to the desired horizontal position but cannot reduce the bouncing height below $z_b \approx 0.3$ m.

B. Control of A Bouncing Ball

This section uses the SAC algorithm with the adjoint variable (40)[22] to develop closed-loop controls on-line that drive a hybrid impulsive bouncing ball model toward different desired states. The system state vector consists of the 2D position and velocity of the ball, $x = (\, x_b,\, z_b,\, \dot{x}_b,\, \dot{z}_b \,)$. The system inputs are constrained accelerations, $u = (\, a_x,\, a_z \,)$: $a_x \in [-10, 10\,]\,\frac{\text{m}}{\text{s}^2}$, $a_z \in [-10, 0\,]\,\frac{\text{m}}{\text{s}^2}$, such that the dynamics are

$$f_q(x,u) = \begin{pmatrix} \dot{x}_b \\ \dot{z}_b \\ a_x \\ a_z - g \end{pmatrix} : \forall q \in \mathcal{Q}\,.$$

As in the previous example, impacts are conservative and so reflect velocity orthogonal to the surface.

The SAC algorithm is initialized from half a meter off the ground, $x_{init} = (\, 0,\, 0.5\,\text{m},\, 0,\, 0\,)$, and results are presented for two different tracking scenarios assuming a flat floor at $z_b = 0\,\text{m}$ as the impact surface (guard). In the first case, SAC uses the quadratic tracking cost (12) with $Q = Diag[\,0,\, 10,\, 0,\, 0\,]$, $P = Diag[\,10,\, 0,\, 0,\, 0\,]$, and applies $R = Diag[\,1,\, 1\,]$ with $T = 0.5\,\text{s}$, $\gamma = -10$, and feedback sampling at $100\,\text{Hz}$.[23]

[22]The first term of $\bar{\rho}$ is always 1 and can be stripped to obtain an unappended hybrid adjoint, ρ, which applies to unappended dynamics as in (4) when the incremental cost does not depend on the control (as in (3)).

[23]The hybrid examples specify SAC with parameters that cause it to skip the (optional) control search process in Section II-B as it is unnecessarily in these cases and complicates analysis.

In this scenario, SAC is set to minimize error between the trajectory of the ball and a desired state a meter to the right of its starting position and one meter above the ground, $x_d = (\, 1\,\text{m},\, 1\,\text{m},\, 0,\, 0\,)$. The $10\,\text{s}$ closed-loop tracking results included in Fig. 13a require $0.21\,\text{s}$ to simulate using the C++ SAC implementation from Section III.

Accelerating the ball in the horizontal directions, SAC drives it the desired horizontal position 1 m away. Due to the control constraints on a_z, however, SAC cannot achieve the desired height. Instead, Fig. 13a shows SAC accelerates the ball into the ground to increase its height after impact. The behavior cannot be achieved without the hybrid modifications to the adjoint variable (40) introduced here. Without the jump terms in the adjoint simulation (from reset map $\Pi_{q,q'}$), the mode insertion gradient (4) does not switch signs at impact events as in Fig. 12 and so does not accurately model the sensitivity to control actions.

A similar demonstration in Fig. 13b shows SAC tracking the desired state, $x_d = (\, 1\,\text{m},\, 0,\, 0,\, 0\,)$, which is also 1 m from the starting position but on the ground. Results take $0.29\,\text{s}$ to compute and are based on all the same parameters previously mentioned but with $Q = Diag[\,0,\, 0,\, 0,\, 10\,]$, so that the cost includes errors on horizontal position (from the P matrix specifying the terminal cost) and vertical velocity.

Because the system is conservative, SAC must act in the a_z direction to remove energy. As SAC can only accelerate the ball into the ground, the algorithm waits until the ball's momentum carries it upward and away from the floor to apply

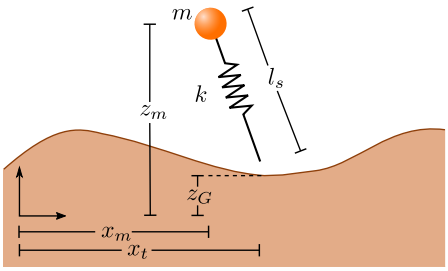


Fig. 14: Planar configuration variables for the SLIP.

control, a_z. Lastly, if one applies the smooth version of SAC from Section II-A to this control scenario, the algorithm will control the ball to the desired horizontal point. While it will reduce the height to approximately $0.3\,\text{m}$, it ceases to make further progress (see Fig. 13b). These findings highlight the fact that (4) provides a poor model for hybrid systems with many switching events.

C. Control of a Spring-Loaded Inverted Pendulum

This final example considers control for the spring-loaded inverted pendulum (SLIP), a hybrid system used to model the dynamics of robotic locomotion. The SLIP is popular because it is a lower dimensional representation of the center of mass dynamics during running for a variety of species [20], [64]. This section uses a 12 dimensional (9 states and 3 controls) model that is similar to the one in [6]. Figure 14 depicts the SLIP's planar configuration variables.

The SLIP's dynamics are divided into flight and stance modes. In our case, the state vector is the same for each mode and includes the 3D position / velocity of the mass, the 2D position of the spring endpoint ("toe"), and a book-keeping variable, $q \in \{q_1, q_2\}$, tracking the current hybrid location (indicating if the SLIP is in flight or stance), $x = (\,x_m, \dot{x}_m, y_m, \dot{y}_m, z_m, \dot{z}_m, x_t, y_t, q\,)$. The control vector is 3 dimensional, $u = (\,u_{t_x}, u_{t_y}, u_{t_y}\,)$, composed of toe velocity controls, which can only be applied in flight, and the leg thrust during stance. The controls are further constrained so the toe velocities are $\in [-5, 5]\,\frac{\text{m}}{\text{s}}$ and $|u_s| \leq 30\,\text{N}$.

Ignoring the location variable, q, the stance dynamics,

$$f_{q_1}(x,u) = \begin{pmatrix} \dot{x}_m \\ \frac{(k(l_0-l_s)+u_s)(x_m-x_t)}{ml_s} \\ \dot{y}_m \\ \frac{(k(l_0-l_s)+u_s)(y_m-y_t)}{ml_s} \\ \dot{z}_m \\ \frac{(k(l_0-l_s)+u_s)(z_m-z_G)}{ml_s} - g \\ 0 \\ 0 \end{pmatrix}, \tag{44}$$

define the first hybrid mode, and flight dynamics,

$$f_{q_2}(x,u) = \big(\,\dot{x}_m,\, 0,\, \dot{y}_m,\, 0,\, \dot{z}_m,\, -g,\, \dot{x}_m + u_{t_x},\, \dot{y}_m + u_{t_y}\,\big), \tag{45}$$

define the second. These dynamics depend on gravity, g, mass, $m = 1\,\text{kg}$, spring constant, $k = 100\frac{\text{N}}{\text{m}}$, the ground height at the toe location, z_g, and the leg length during stance,

$$l_s = \sqrt{(x_m - x_t)^2 + (y_m - y_t)^2 + (z_m - z_G)^2}. \tag{46}$$

When $l_s = l_0$, the guard equations,

$$\phi_{q_2,q_1}(x) = \phi_{q_1,q_2}(x) = z_m - \frac{l_0(z_m - z_G)}{l_s} - z_G\,, \tag{47}$$

cross zero to indicating the transition from stance to flight mode (and vice versa). Upon transitioning to flight, the leg length becomes fixed at the resting length, $l_0 = 1\,\text{m}$. Reset maps Ω_{q_2,q_1} and Ω_{q_1,q_2} leave the state unchanged other than to update the location variable, q.

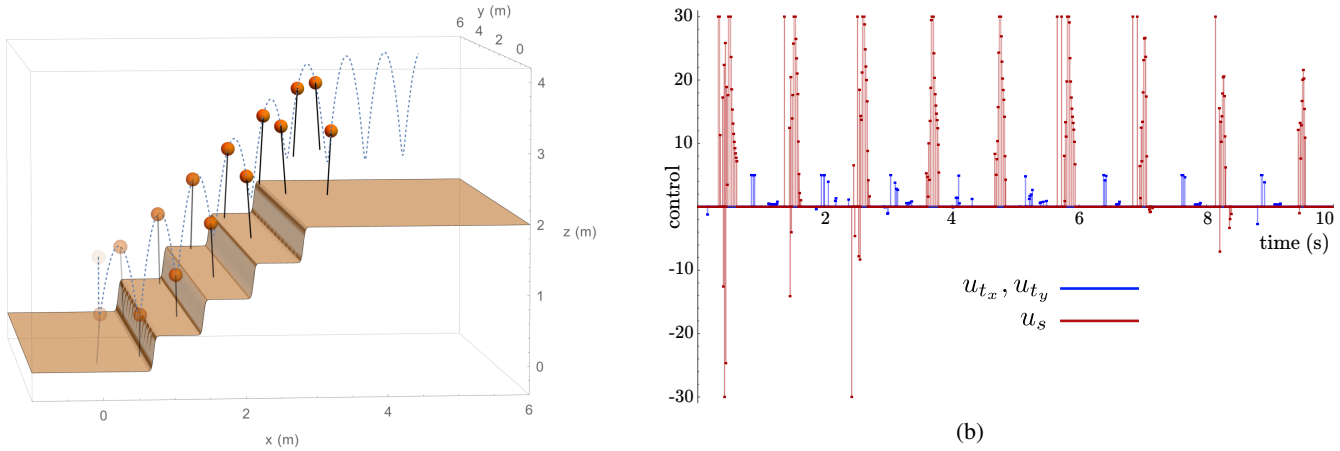
Figure 15 includes a sample trajectory based on a quadratic objective with $Q = Diag[\,0\,,70\,,0\,,70\,,50\,,0\,,0\,,0\,]$, $R = I$, $P_1 = 0$, $T = 0.6\,\text{s}$, and $\alpha_d = -10$. The figure depicts SAC controlling the SLIP up a staircase, which is approximating using logistic functions, $z_g = \sum_{i=1}^{4} \frac{0.5}{1 + e^{-75(x-0.7i)}}$. These functions produce stairs with a slope of ≈ 0.71 (a $0.5\,\text{m}$ rise every $0.7\,\text{m}$). As the rise of each step is equal to half the SLIP body length, SAC must coordinate leg motion to avoid tripping on the stair ledges. With the desired trajectory, $x_d = (\,0\,, 0.7\,\frac{\text{m}}{\text{s}}\,, 0\,, 0.7\,\frac{\text{m}}{\text{s}}\,, z_G + 1.4\,\text{m}\,, 0\,, 0\,, 0\,)$, SAC drives the SLIP along a diagonal path up the staircase at roughly constant velocity and relatively uniform average height above the ground. The $10\,\text{s}$ trajectory simulates in $\approx 1.6\,\text{s}$ on a laptop with feedback at 100 Hz.[24] The hybrid SAC controller successfully navigates the SLIP over a variety of other terrain types, including sloped sinusoidal floors, using these same parameters and with similar timing results. More recent results confirm SAC also extends to two legged compliant walking models from [20]. Both these varied terrain SLIP locomotion and compliant walking examples are in the video attachment.

The SLIP is well-studied, and researchers have already derived methods that yield stable hopping by controlling the SLIP leg to desired touchdown angles. These methods typically assume the leg can swing arbitrarily fast to implement analytically computed touchdown angles at each step and ignore possible collisions with terrain during swing. This example shows that the hybrid version of SAC can drive the SLIP over varying terrain while controlling the motion of the leg to avoid tripping. We note that SAC implementations like the one introduced here may prove useful in controlling robots that (mechanically) emulate the SLIP [13], [28], [53]. Due to physical constraints, these robots are limited in how well they can approximate the SLIP model assumptions and so SLIP-based control may prove ineffective. In contrast, SAC can be applied to the actual robot model (or a more accurate model) and does not rely on the simplifying SLIP assumptions to control locomotion.

VI. Conclusions and Future Work

This paper contributes a model-based algorithm, Sequential Action Control (SAC), that sequences optimal actions into a closed-loop, trajectory-constrained control at roughly the rate of simulation. While the approach is new and further study is required to define properties like robustness and sensitivities, we have tested SAC on an array of problems spanning several

[24] The process is artificially slowed by impact event detection code, which we are still developing.



(a) (b)

Fig. 15: A time lapse showing the SLIP at $0.5\,\mathrm{s}$ increments (Fig. 15a) under SAC controls (Fig. 15b).

categories of traditionally challenging system types. These benchmark trials confirm the algorithm can outperform standard methods for nonlinear optimal control and case specific controllers in terms of tracking performance and speed.

For the continued development of SAC, a number of directions have been identified as possible future work. For instance, although we show SAC can avoid local minima that affect nonlinear trajectory optimization, the method is local in the sense that it cannot guarantee globally optimal solutions through state space (no method can in finite time for the nonlinear / non-convex problems here). As such, despite the wide range of systems SAC can control, there are others that will prove difficult. To increase applicability, SAC can be combined with global, sample-based planners to provide a fast local planner that develops constrained solutions that satisfy dynamics. Such methods would allow SAC to apply even in very complicated scenarios such as those required to develop trajectories for humanoids [24], [66].

To better automate control policy generation and reduce required user input, SAC needs tools for parameter tuning, especially ones that provide stability. As mentioned in Section II, SAC parameters can be selected to provide local stability around equilibrium based on a linear state feedback law for optimal actions (13). Sums-of-Squares (SOS) tools (e.g., the S-procedure) [49], [68] seems a good candidate to automate parameter tuning and the generation of regions of attraction.

In addition to the applications mentioned, we note that SAC applies broadly to auto-pilot and stabilization systems like those in rockets, jets, helicopters, autonomous vehicles, and walking robots [18], [24], [45], [57], [70]. It also naturally lends itself to shared control systems where the exchange between human and computer control can occur rapidly (e.g., wearable robotics and exoskeletons [23], [30], [33], [43], [76]). It offers a reactive, on-line control process that can reduce complexity and pre-computation required for robotic perching and aviation experiments in [8], [9], [68]. Its speed may facilitate predictive feedback control for new flexible robots [37], [58] and systems that are currently restricted to open-loop. It offers real-time system ID and parameter estimation for nonlinear systems [51], [74], [75]. These potential applications merit study and further development of the SAC approach.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant CMMI 1200321. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Thamer Albahkali, Ranjan Mukherjee, and Tuhin Das. Swing-up control of the pendubot: an impulse–momentum approach. *IEEE Transactions on Robotics*, 25(4):975–982, 2009.

[2] Frank Allgower, Rolf Findeisen, and Zoltan K Nagy. Nonlinear model predictive control: From theory to application. *Journal of the Chinese Institute of Chemical Engineers*, 35(3):299–316, 2004.

[3] Frank Allgöwer and Alex Zheng. *Nonlinear model predictive control*, volume 26. Birkhäuser Basel, 2000.

[4] Brian D. O. Anderson and John B. Moore. *Optimal control: linear quadratic methods*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1990.

[5] Alex R Ansari and Todd D Murphey. Control-on-request: Short-burst assistive control for long time horizon improvement. In *American Control Conference*, 2015.

[6] Omür Arslan. *Model based methods for the control and planning of running robots*. PhD thesis, Bilkent University, 2009.

[7] Karl Johan Åström and Katsuhisa Furuta. Swinging up a pendulum by energy control. *Automatica*, 36(2):287–295, 2000.

[8] Andrew J Barry. Flying between obstacles with an autonomous knife-edge maneuver. Master's thesis, MIT, 2012.

[9] Andrew J. Barry, Tim Jenks, Anirudha Majumdar, Huai-Ti Lin, Ivo G. Ros, Andrew Biewener, and Russ Tedrake. Flying between obstacles with an autonomous knife-edge maneuver. In *IEEE Conference on Robotics and Automation*, Video Track, 2014.

[10] Anthony M Bloch, Melvin Leok, Jerrold E Marsden, and Dmitry V Zenkov. Controlled lagrangians and stabilization of the discrete cart-pendulum system. In *IEEE Conference on Decision and Control (CDC) and the European Control Conference (ECC)*, pages 6579–6584, 2005.

[11] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[12] Tim M Caldwell and Todd D Murphey. Projection-based optimal mode scheduling. In *IEEE Conference on Decision and Control*, 2013.

[13] B. Dadashzadeh, H.R. Vejdani, and J. Hurst. From template to anchor: A novel control strategy for spring-mass running of bipedal robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2566–2571, Sept 2014.

[14] Moritz Diehl, Rishi Amrit, and James B Rawlings. A Lyapunov function for economic optimizing model predictive control. *IEEE Transactions on Automatic Control*, 56(3):703–707, 2011.

[15] Magnus Egerstedt, Yorai Wardi, and Henrik Axelsson. Optimal control of switching times in hybrid systems. In *International Conference on Methods and Models in Automation and Robotics*, 2003.

[16] Magnus Egerstedt, Yorai Wardi, and Henrik Axelsson. Transition-time optimization for switched-mode dynamical systems. *IEEE Transactions on Automatic Control*, 51(1):110–115, 2006.

[17] Brian C Fabien. Implementation of a robust SQP algorithm. *Optimization Methods & Software*, 23(6):827–846, 2008.

[18] Paolo Falcone, Francesco Borrelli, Jahan Asgari, H Eric Tseng, and Davor Hrovat. Predictive active steering control for autonomous vehicle systems. *IEEE Transactions on Control Systems Technology*, 15(3):566–580, 2007.

[19] Isabelle Fantoni, Rogelio Lozano, and Mark W Spong. Energy based control of the pendubot. *IEEE Transactions on Automatic Control*, 45(4):725–729, 2000.

[20] Hartmut Geyer, Andre Seyfarth, and Reinhard Blickhan. Compliant leg behaviour explains basic dynamics of walking and running. *Proceedings of the Royal Society B: Biological Sciences*, 273(1603):2861–2867, 2006.

[21] Philip E Gill, Walter Murray, and Michael A Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM journal on optimization*, 12(4):979–1006, 2002.

[22] Humberto Gonzalez, Ram Vasudevan, Maryam Kamgarpour, S Shankar Sastry, Ruzena Bajcsy, and Claire J Tomlin. A descent algorithm for the optimal control of constrained nonlinear switched dynamical systems. In *ACM Conference on Hybrid Systems: Computation and Control*, pages 51–60, 2010.

[23] R.D. Gregg, T.W. Bretl, and M.W. Spong. A control theoretic approach to robot-assisted locomotor therapy. In *IEEE Conference on Decision and Control (CDC)*, pages 1679–1686, Dec 2010.

[24] Robert D Gregg, Adam K Tilton, Salvatore Candido, Timothy Bretl, and Mark W Spong. Control and planning of 3-d dynamic walking with asymptotically stable gait primitives. *IEEE Transactions on Robotics*, 28(6):1415–1423, 2012.

[25] Lars Grüne and Jürgen Pannek. *Nonlinear model predictive control*. Springer, 2011.

[26] John Hauser. "A projection operator approach to the optimization of trajectory functionals". In *IFAC World Congress*, Barcelona, Spain, Jul. 2002.

[27] João P Hespanha. *Linear systems theory*. Princeton university press, 2009.

[28] Jonathan W Hurst, Joel E Chestnutt, and Alfred A Rizzi. The actuator with mechanically adjustable series compliance. *IEEE Transactions on Robotics*, 26(4):597–606, 2010.

[29] Ali Jadbabaie and John Hauser. On the stability of receding horizon control with a general terminal cost. *IEEE Transactions on Automatic Control*, 50(5):674–678, 2005.

[30] Saso Jezernik, Gery Colombo, and Manfred Morari. Automatic gait-pattern adaptation algorithms for rehabilitation with a 4-dof robotic orthosis. *IEEE Transactions on Robotics and Automation*, 20(3):574–582, 2004.

[31] T.A Johansen, T.I Fossen, and S.P. Berge. Constrained nonlinear control allocation with singularity avoidance using sequential quadratic programming. *IEEE Transactions on Control Systems Technology*, 12(1):211–216, Jan 2004.

[32] Elliot R. Johnson and Todd D. Murphey. Scalable Variational Integrators for Constrained Mechanical Systems in Generalized Coordinates. *IEEE Transactions on Robotics*, 25(6):1249–1261, 2009.

[33] Hami Kazerooni, Andrew Chu, and Ryan Steger. That which does not stabilize, will only make us stronger. *The International Journal of Robotics Research*, 26(1):75–89, 2007.

[34] Hassan K. Khalil and J.W. Grizzle. *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, 2002.

[35] Paul Kulchenko and Emanuel Todorov. First-exit model predictive control of fast discontinuous dynamics: Application to ball bouncing. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2144–2151, 2011.

[36] Xu-Zhi Lai, Jin-Hua She, Simon X Yang, and Min Wu. Comprehensive unified control strategy for underactuated two-link manipulators. *Systems, Man, and Cybernetics, Part B: IEEE Transactions on Cybernetics*, 39(2):389–398, 2009.

[37] Cecilia Laschi, Matteo Cianchetti, Barbara Mazzolai, Laura Margheri, Maurizio Follador, and Paolo Dario. Soft robot arm inspired by the octopus. *Advanced Robotics*, 26(7):709–727, 2012.

[38] Jay H Lee. Model predictive control: review of the three decades of development. *International Journal of Control, Automation and Systems*, 9(3):415–424, 2011.

[39] Sven Leyffer and Ashutosh Mahajan. Software for nonlinearly constrained optimization. *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

[40] Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, pages 222–229, 2004.

[41] Daniel Liberzon. *Calculus of variations and optimal control theory: a concise introduction*. Princeton University Press, 2012.

[42] Rogelio Lozano, Isabelle Fantoni, and Dan J Block. Stabilization of the inverted pendulum around its homoclinic orbit. *Systems & control letters*, 40(3):197–204, 2000.

[43] Anastasia Mavrommati, Alex R Ansari, and Todd D Murphey. Optimal control-on-request: An application in real-time assistive balance control. In *ICRA workshop on open source software*, 2015.

[44] David Q Mayne, James B Rawlings, Christopher V Rao, and Pierre OM Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[45] Luis Mejias, Srikanth Saripalli, Pascual Campoy, and Gaurav S Sukhatme. Visual servoing of an autonomous helicopter in urban areas using feature tracking. *Journal of Field Robotics*, 23(3-4):185–199, 2006.

[46] Nenad Muskinja and Boris Tovornik. Swinging up and stabilization of a real inverted pendulum. *IEEE Transactions on Industrial Electronics*, 53(2):631–639, 2006.

[47] D Subbaram Naidu. *Optimal control systems*, volume 2. CRC press, 2002.

[48] Yury Orlov, Luis T Aguilar, Leonardo Acho, and Adain Ortiz. Swing up and balancing control of pendubot via model orbit stabilization: Algorithm synthesis and experimental verification. In *45th IEEE Conference on Decision and Control*, pages 6138–6143. IEEE, 2006.

[49] Pablo A Parrilo. *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*. PhD thesis, California Institute of Technology, 2005.

[50] L.S. Pontryagin, V.G. Boltyanskii, R.V. Gamkrelidze, and E.F. Mischenko. *The mathematical theory of optimal processes,* K.N. Trirogoff (transl.), L.W. Neustadt (ed.). Interscience Publishers, New York, 1962.

[51] Luc Pronzato. Optimal experimental design and some related control problems. *Automatica*, 44(2):303–325, 2008.

[52] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, volume 3, 2009.

[53] Marc H Raibert. Legged robots. *Communications of the ACM*, 29(6):499–514, 1986.

[54] Anil V Rao. A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, 135(1):497–528, 2009.

[55] Klaus Schittkowski. NLPQLP: A Fortran implementation of a sequential quadratic programming algorithm with distributed and non-monotone line search-users guide, version 2.2, 2006.

[56] Klaus Schittkowski. A robust implementation of a sequential quadratic programming algorithm with successive error restoration. *Optimization Letters*, 5(2):283–296, 2011.

[57] R.B. Schroer. Flight control goes digital [part two, NASA at 50]. *Aerospace and Electronic Systems Magazine, IEEE*, 23(10):23–28, 2008.

[58] Robert F Shepherd, Filip Ilievski, Wonjae Choi, Stephen A Morin, Adam A Stokes, Aaron D Mazzeo, Xin Chen, Michael Wang, and George M Whitesides. Multigait soft robot. *Proceedings of the National Academy of Sciences*, 108(51):20400–20403, 2011.

[59] Mark W Spong. The swing up control problem for the acrobot. *IEEE Control Systems*, 15(1):49–55, 1995.

[60] Mark W Spong. Underactuated mechanical systems. In *Control Problems in Robotics and Automation*, pages 135–150. Springer, 1998.

[61] Mark W Spong and Daniel J Block. The pendubot: A mechatronic system for control research and education. In *IEEE Conference on Decision and Control*, volume 1, pages 555–556, 1995.

[62] Mark W Spong, Peter Corke, and Rogelio Lozano. Nonlinear control of the reaction wheel pendulum. *Automatica*, 37(11):1845–1851, 2001.

[63] B Srinivasan, P Huguenin, and Dominique Bonvin. Global stabilization of an inverted pendulum–control strategy and experimental verification. *Automatica*, 45(1):265–269, 2009.

[64] Manoj Srinivasan and Andy Ruina. Computer optimization of a minimal biped model discovers walking and running. *Nature*, 439(7072):72–75, 2005.
[65] Héctor J Sussmann. A maximum principle for hybrid optimal control problems. In *IEEE Conference on Decision and Control*, volume 1, pages 425–430, 1999.
[66] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4906–4913, 2012.
[67] Yuval Tassa, Nicolas Mansard, and Emo Todorov. Control-limited differential dynamic programming. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1168–1175. IEEE, 2014.
[68] Russ Tedrake, Ian R Manchester, Mark Tobenkin, and John W Roberts. LQR-trees: Feedback motion planning via sums-of-squares verification. *The International Journal of Robotics Research*, 2010.
[69] Emanuel Todorov and Weiwei Li. A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *American Control Conference (ACC)*, pages 300–306, 2005.
[70] Matthew Turpin, Nathan Michael, and Vijay Kumar. Trajectory design and control for aggressive formation flight with quadrotors. *Autonomous Robots*, 33(1-2):143–156, 2012.
[71] Richard Vinter. *Optimal control*. Springer, 2010.
[72] Yorai Wardi, M Egerstedt, and P Twu. A controlled-precision algorithm for mode-switching optimization. In *IEEE Conference on Decision and Control (CDC)*, pages 713–718, 2012.
[73] Yorai Wardi and Magnus Egerstedt. Algorithm for optimal mode scheduling in switched systems. In *American Control Conference*, pages 4546–4551, 2012.
[74] Andrew D Wilson, J.A. Schultz, and Todd D Murphey. Trajectory optimization for well-conditioned parameter estimation. *IEEE Transactions on Automation Science and Engineering*, 12(1):28–36, 2015.
[75] Andrew D Wilson, Jarvis A Schultz, Alex R Ansari, and Todd D Murphey. Real-time trajectory synthesis for information maximization using sequential action control and least-squares estimation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.
[76] Kyle N Winfree, Paul Stegall, and Sunil K Agrawal. Design of a minimally constraining, passively supported gait training exoskeleton: ALEX II. In *IEEE International Conference on Rehabilitation Robotics (ICORR)*, pages 1–6, 2011.
[77] SJ Wright and J Nocedal. *Numerical optimization*, volume 2. Springer New York, 1999.
[78] X Xin and M Kaneda. Analysis of the energy-based swing-up control of the acrobot. *International Journal of Robust and Nonlinear Control*, 17(16):1503–1524, 2007.
[79] Xin Xin and Taiga Yamasaki. Energy-based swing-up control for a remotely driven acrobot: Theoretical and experimental results. *IEEE Transactions on Control Systems Technology*, 20(4):1048–1056, 2012.
[80] Ji-Hyuk Yang, Su-Yong Shim, Jung-Hun Seo, and Young-Sam Lee. Swing-up control for an inverted pendulum with restricted cart rail length. *International Journal of Control, Automation and Systems*, 7(4):674–680, 2009.

APPENDIX

A. Input Constraints

This section provides several means to incorporate min-max saturation constraints on elements of the optimal action vector. To simplify the discussion and analysis presented, it is assumed that the nominal control vector $u_1 = 0$, as in the implementation examples.

1) Control Saturation – Quadratic Programming: While more efficient alternatives will be presented subsequently, the most general way to develop controls that obey saturation constraints is by minimizing (48) subject to inequality constraints. The following proposition provides the resulting quadratic programming problem in the case of $u_1 = 0$.

Proposition 1. *At any application time $s = \tau_{m,i}$, a control action exists that obeys saturation constraints from the constrained quadratic programming problem*

$$u_2^*(s) = \arg\min_{u_2(s)} \frac{1}{2}\|\Gamma(s)\, u_2(s) - \alpha_d\|^2 + \frac{1}{2}\|u_2(s)\|_R^2 \quad (48)$$

such that $u_{min,k} \leq u_{2,k}^*(s) \leq u_{max,k}\ \forall k \in \{1, \ldots, m\}$. The term $\Gamma^T \triangleq h(x)^T \rho$, and values $u_{min,k}$ and $u_{max,k}$ bound the k^{th} component of $u_2^*(s)$.

Proof: For control-affine systems with the null vector as the nominal control, the mode insertion gradient (4) simplifies to

$$\begin{aligned}\frac{dJ_1}{d\lambda_i^+} &= \rho(s)^T \left(f_2(s,s) - f_1(s)\right) \\ &= \rho(s)^T\, h(x(s))\, u_2^*(s) \\ &= \langle\, \Gamma(s)^T, u_2^*(s)\,\rangle\,. \end{aligned} \quad (49)$$

The final relation is stated in terms of an inner product.

With the linear mode insertion gradient (49), minimizing (48) subject to $u_{min,k} \leq u_{2,k}^*(s) \leq u_{max,k}\ \forall k$ is equivalent to optimizing (6) at time s to find a saturated action, $u_2^*(s)$. ■

Proposition 1 considers a constrained optimal action, $u_2^*(s)$, at a fixed time. However, the quadratic programming approach can be used to search for the schedule of solutions u_2^* that obey saturation constraints (though it would increase computational cost). These quadratic programming problems can be solved much more efficiently than the nonlinear dynamics constrained programming problems that result when searching for finite duration optimal control solutions. As described next, even the limited overhead imposed by these problems can be avoided by taking advantage of linearity in (8).

2) Control Saturation – Vector Scaling: Optimal actions computed from (8) are affine with respect to α_d and linear when $u_1 = 0$. Thus, scaling α_d to attempt more dramatic changes in cost relative to control duration produces actions that are scaled equivalently.[25] Consider the k^{th} component of a control action vector, $u_{2,k}^*(s)$, has minimum and maximum saturation constraints encoded by the k^{th} component of saturation vectors of the form $u_{min,k} < 0 < u_{max,k}\ \forall k \in \{1, \ldots, m\}$. The linear relationship between $u_2^*(s)$ and α_d implies that if any component $u_{2,k}^*(s) > u_{max,k}$ or $u_{2,k}^*(s) < u_{min,k}$, one can choose a new $\hat{\alpha}_d$ that positively scales the entire control vector until constraints are satisfied. If the worst constraint violation is due to a component $u_{2,k}^*(s) > u_{max,k}$, choosing $\hat{\alpha}_d = \alpha_d\, u_{max,k}/u_{2,k}^*(s)$ will produce a positively scaled $u_2^*(s)$ that obeys all constraints. Linearity between $u_2^*(s)$ and α_d implies that this factor can be directly applied to control actions from (8) rather than re-calculating from $\hat{\alpha}_d$. To guarantee that scaling control vectors successfully returns solutions that obey constraints and reduce cost (3), constraints must be of the form $u_{min,k} < 0 < u_{max,k}\ \forall k$.

Proposition 2. *For the choice $\alpha_d < 0$, a control action $u_2^*(s)$ evaluated anywhere that $\Gamma(s)^T \triangleq h(x(s))^T\, \rho(s) \neq 0 \in \mathbb{R}^{m\times 1}$ will result in a negative mode insertion gradient* (4) *and so can reduce* (3).

[25] Generally, scaling α_d will not equivalently scale the overall change in cost because the neighborhood, V_i, where the (18) models the change in cost can change. This would result in a different duration λ_i for the scaled action.

Proof: Combining (8) with (49), optimal actions that reduce cost result in a mode insertion gradient satisfying

$$\begin{aligned}\frac{dJ_1}{d\lambda_i^+} &= \langle\, \Gamma(s)^T, (\Gamma(s)^T\Gamma(s) + R^T)^{-1}\,\Gamma(s)^T\alpha_d\,\rangle \\ &= \alpha_d\,\|\Gamma(s)^T\|^2_{(\Gamma(s)^T\Gamma(s)+R^T)^{-1}} < 0\,.\end{aligned}$$

The outer product, $\Gamma(s)^T\Gamma(s)$, produces a positive semi-definite symmetric matrix. Adding $R > 0$ yields a positive definite matrix. Because the inverse of a positive definite matrix is positive definite, the quadratic norm $\|\Gamma(s)^T\|^2_{(\Gamma(s)^T\Gamma(s)+R^T)^{-1}} > 0$ for $\Gamma(s)^T \neq 0 \in \mathbb{R}^{m\times 1}$. Therefore, only choices $\alpha_d < 0$ in (9) produce optimal control actions that make $\frac{dJ_1}{d\lambda_i^+} < 0$ and by (18) can reduce cost (3). ■

3) Control Saturation – Element Scaling: For multidimensional vectors, scaling can produce overly conservative (unnecessarily small magnitude) controls when only a single vector component violates a constraint. To avoid the issue and reap the computational benefits of vector scaling, one can choose to scale the individual components of a multi-dimensional action, $u_2^*(s)$, by separate factors to provide admissible solutions (saturated control actions that reduce (3)). The following proposition presents conditions under which this type of saturation guarantees admissible controls.

Proposition 3. *Assume $R = c\,I$ where I is the identity and $c \in \mathbb{R}^+$, $\alpha_d \in \mathbb{R}^-$, $u_1 = 0$, and separate saturation constraints $u_{min,k} \leq 0 \leq u_{max,k}\ \forall k \in \{1,\ldots,m\}$ apply to elements of the control vector. The components of any control derived from* (8) *and evaluated at any time, s, where $\Gamma(s)^T \triangleq h(x(s))^T\,\rho(s) \neq 0 \in \mathbb{R}^{m\times 1}$ can be independently saturated. If $\|u_2^*(s)\| \neq 0$ after saturation, the action is guaranteed to be capable of reducing cost* (3)*.*

Proof: For the assumptions stated in Proposition 3,

$$u_2^*(s) = (\Gamma(s)^T\Gamma(s) + R^T)^{-1}\,\Gamma(s)^T\alpha_d\,.$$

The outer product, $\Gamma(s)^T\Gamma(s)$, produces a rank 1 positive semi-definite, symmetric matrix with non-zero eigenvalue $= \Gamma(s)\Gamma(s)^T$ associated with eigenvector $\Gamma(s)^T$. Eigenvalue decomposition of the outer product yields $\Gamma(s)^T\Gamma(s) = S\,D\,S^{-1}$, where the columns of S corresponds to the eigenvectors of $\Gamma(s)^T\Gamma(s)$ and D is a diagonal matrix of eigenvalues. For $R = R^T = c\,I$, actions satisfy

$$\begin{aligned}u_2^*(s) &= (S\,D\,S^{-1} + c\,I)^{-1}\,\Gamma(s)^T\alpha_d \\ &= (S\,D\,S^{-1} + c\,S\,I\,S^{-1})^{-1}\,\Gamma(s)^T\alpha_d \\ &= (S\,(D + c\,I)\,S^{-1})^{-1}\,\Gamma(s)^T\alpha_d \\ &= S\,(D + c\,I)^{-1}\,S^{-1}\,\Gamma(s)^T\alpha_d\,.\end{aligned}$$

The matrix $D + c\,I$ in the final statement must be symmetric and positive-definite with eigenvalues all equal to c except for the one associated with the nonzero eigenvalue of D. This eigenvalue, $\Gamma(s)\Gamma(s)^T + c$, applies to eigenvectors that are scalar multiples of $\Gamma(s)^T$. After inversion, matrix $S\,(D + c\,I)^{-1}\,S^{-1}$ must then have an eigenvalue $\frac{1}{\Gamma(s)\Gamma(s)^T + c}$. Since inversion of a diagonal matrix leaves its eigenvectors unchanged, the eigenvalue scales $\Gamma(s)^T$. Therefore, the matrix $S\,(D + c\,I)^{-1}\,S^{-1}$ directly scales its eigenvector, $\Gamma(s)^T$, and

$$u_2^*(s) = \frac{\alpha_d}{\Gamma(s)\Gamma(s)^T + c}\,\Gamma(s)^T\,. \tag{50}$$

For any $\alpha_d \in \mathbb{R}^-$, $u_2^*(s)$ will be a negative scalar multiple of $\Gamma(s)^T$. Because two vectors $\in \mathbb{R}^m$ can at most span a $2D$ plane $E \subset \mathbb{R}^m$, the Law of Cosines (the angle, ϕ, between vectors u and v can be computed from $\cos(\phi) = \frac{\langle u,v\rangle}{\|u\|\|v\|}$) can be applied to compute the angle between any $u_2^*(s)$ and $\Gamma(s)^T$. The Law of Cosines verifies that control (50) is 180° relative to $\Gamma(s)^T$. Therefore, (50) corresponds to the control of least Euclidean norm that minimizes (49) and so maximizes the expected change in cost. The Law of Cosines and (49) also imply the existence of a hyperplane, $h_p := \{\nu(s) \in \mathbb{R}^m \mid \langle\, \Gamma(s)^T, \nu(s)\,\rangle = 0\}$, of control actions, $\nu(s)$, orthogonal to both (50) and $\Gamma(s)^T$. This hyperplane divides R^m into subspaces composed of vectors capable of reducing cost (3) (they produce a negative mode insertion gradient based on inner product (49)) and those that cannot.

To show that saturation returns a vector in the same subspace as (50), one can define the control in terms of component magnitudes, $a = (a_1,\ldots,a_m)$, and signed orthonormal bases from $\mathbb{R}^m$, $\hat{e} = (\hat{e}_1,\ldots,\hat{e}_m)$, so that $u_2^*(s) = a\,\hat{e}$. The Law of Cosines confirms that $u_2^*(s)$ can be written only in terms of components a_k and signed basis vectors $\hat{e}_k$ within acute angles of the control. Briefly, the law indicates an a_k cannot be associated with any basis, $\hat{e}_k$, at 90° of the control because it would require $\langle\, \hat{e}_i, u_2^*(s)\,\rangle = 0$, implying $a_k = 0$. Similarly, an a_k cannot be associated with an $\hat{e}_k > 90^\circ$ relative to the control because this is equivalent to $\langle\, \hat{e}_k, u_2^*(s)\,\rangle < 0$, and leads to an $a_k < 0$ that contradicts definition.

Because (50) is represented by positively scaled bases within 90° of $u_2^*(s)$, all these vectors must lie on the same side of h_p as (50). This is also true of any vector produced by a non-negative linear combination of the components of $u_2^*(s)$. Since there always exists factors $\in [0,\infty)$, that can scale the elements of an action vector until they obey constraints $u_{min,k} \leq 0 \leq u_{max,k}\ \forall k \in \{1,\ldots,m\}$, saturated versions of (50) will still be capable of reducing cost for $\|u_2(s)\| \neq 0$. ■