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

MODEL-BASED control techniques like dynamic programming or trajectory optimization can generate highly efficient motions that leverage, rather than fight, the dynamics of robotic mechanisms. Yet, these tools are often difficult to apply since even basic robot locomotion and manipulation tasks often yield complex nonlinear, hybrid, underactuated, and high-dimensional constrained control problems. For instance, consider the spring-loaded inverted pendulum (SLIP) model in Fig. 1. With only a point-mass body and spring leg, the model provides one of the most idealized locomotion templates that remains popular in robotics for capturing the center of mass (COM) running dynamics of a wide range of species [1], [2]. However, even this simple model exhibits nonlinear, underactuated, and hybrid phenomena that affect optimizations in most model-based methods.

For example, a trajectory optimization approach could derive the dashed solution in Fig. 1, but would usually require a good initial guess (and special care to account for discontinuities). Initialization is important because the nonlinear dynamics imply the constrained objective is non-convex and subject to potentially poor local minima. An additional regulating controller, capable of tracking the trajectory through impacts, would also be necessary to track the resulting solution. Still, the approach would not be able to adapt the trajectory to accommodate a dynamically changing environment.

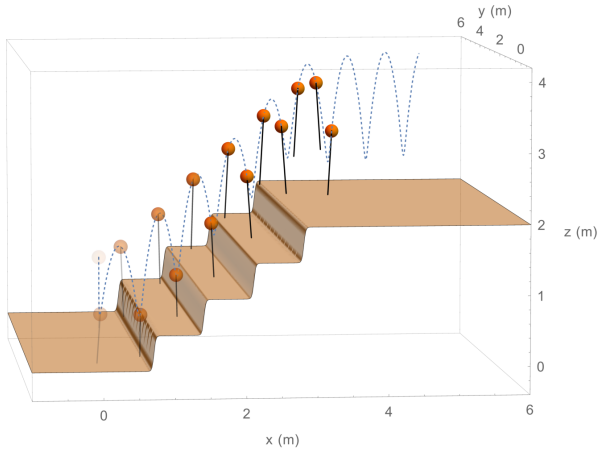


Fig. 1. Time lapse (0.5s) of a spring-loaded inverted pendulum (SLIP) reactively hopping up stairs using SAC.

As an alternative, a (nonlinear) receding horizon control approach would compute an optimal trajectory, follow it for a single time step, and then iterate to construct a closed-loop response. However, each receding horizon problem would still be non-convex, requiring computationally expensive iterative optimization [3]. In addition to local minima issues, the approach would be limited to lower bandwidth scenarios with slowly-varying environmental/dynamic conditions [3], [4].

To address these problems, this paper proposes a new model-based algorithm, which we refer to as *Sequential Action Control* (SAC), that makes strategic trade-offs for computational gain and improved generality. That is, rather than solving for full control curves that minimize a non-convex objective over each receding horizon, SAC finds a single optimal control *value* and *time* to act that maximally *improves* performance.[1] For instance, in the SLIP example SAC waits until the flight phase to tilt the leg backwards, predicting that the action will drive the robot farther up the steps (after the leg re-contacts) for some specified horizon. As in receding

Manuscript received October 27, 2015; revised June 4, 2016; accepted July 9, 2015.

A. Ansari is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: aansari1@andrew.cmu.edu).

T. Murphey is with the Department of Mechanical Engineering, Northwestern University, Evanston, IL 60208 (e-mail: t-murphey@northwestern.edu).

[1]SAC also uses a line search [5] to specify a short duration (usually a single discrete time-step) to apply each control and improve performance.

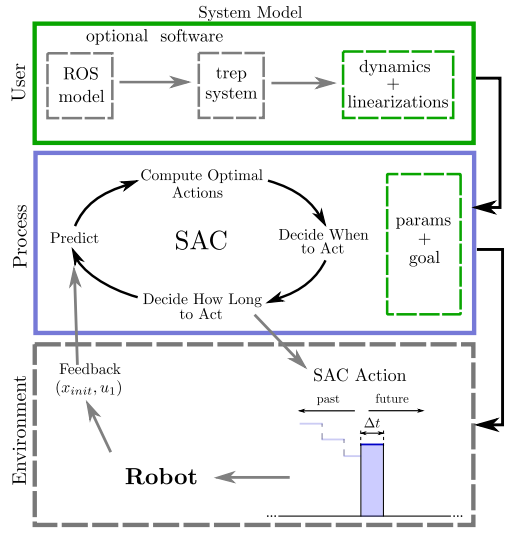


Fig. 2. An overview of the SAC control process including possible open-source interfaces, e.g., ROS [7] and trep [8].

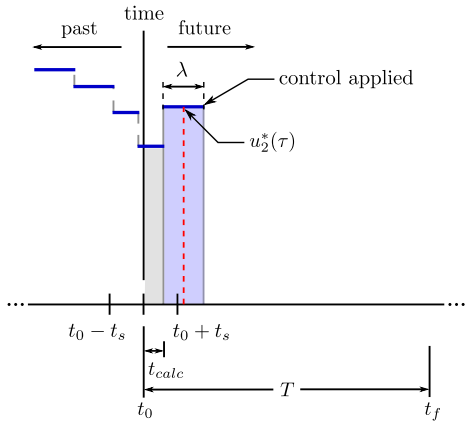


Fig. 3. Following the cyclic process in Fig. 2, SAC computes a *schedule*, $u_2^* : (t_0, t_f) \mapsto \mathbb{R}^m$, providing the value of optimal actions that maximally improve a tracking objective over the current (receding) horizon. Next, SAC selects an application time, $\tau \in (t_0 + t_{calc}, t_f)$, and $u_2^*(\tau)$ becomes the value of the next SAC action (blue shaded bar). A line search sets the duration, λ. Previously computed actions are applied while current calculations complete, $t \in [t_0, t_0 + t_{calc}]$. After incorporating feedback, SAC repeats the process at the next sample time, $t = t_0 + t_s$.

horizon control, SAC incorporates feedback and repeats these calculations at each time step as the horizon recedes. The resulting process computes a real-time, closed-loop response that reacts to terrain and continually drives the robot up the steps. Figure 2 provides an overview of the SAC process.

There are several advantages to the trade-offs SAC makes, i.e., computing individual control actions at each time step that improve performance rather than curves that directly optimize a performance objective. These advantages include: 1) SAC controls can be rapidly computed on-line from a closed-form expression with guaranteed optimality, existence, and uniqueness.[2] 2) SAC controls can be directly saturated to obey min-max constraints without any computational overhead or specialized solvers. 3) SAC's control synthesis process is unaffected by discontinuities in dynamics and so applies to challenging hybrid and impulsive robots. 4) In spite of sacrificing the multi-step planning process of trajectory optimization, benchmark examples demonstrate a final, unintuitive finding – SAC can avoid poor local minima that trap nonlinear optimal control. To illustrate this last point, Section IV includes a number of robotics-related control examples that show SAC outperforms case-specific methods and popular optimal control algorithms (sequential quadratic programming [5] and iLQG [6]). Compared to these alternatives, SAC computes high-bandwidth (1 KHz) closed-loop trajectories with equivalent or better final cost in less time (milliseconds/seconds vs. hours).

To sum up, SAC provides a model-based control response to state that is easily implemented and efficiently computed for for most robotic systems, including those that are saturated, underactuated, nonlinear, hybrid/impulsive, and high dimensional. This paper introduces SAC in two parts. Part I focuses on robots with differentiable nonlinear dynamics, and Part II considers hybrid impulsive robots. Benchmark examples and relevant background material are introduced in the context of each. Table I includes notation used throughout this paper.

[2]There are algorithms other than SAC that yield controls in closed form. However, we are unaware of any methods (particularly model/optimization-based methods) that provide comparable constrained closed-form controls on-line for examples such as those in Section IV and accommodate nonlinear hybrid/impulsive robots.

PART I: SAC FOR DIFFERENTIABLE SYSTEMS

II. CONTROL SYNTHESIS

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)

The subsequent sections detail SAC control synthesis, following the cyclic process in Fig. 2. We describe how each cycle of the SAC process computes an optimal *action – defined by the triplet consisting of a control's value, $u \in \mathbb{R}^m$, a short application duration, $\lambda \in \mathbb{R}^+$, and application time, $\tau \in \mathbb{R}^+$ (see the blue shaded bar in Fig. 3)* – that is sent to a robot.

A. Prediction

At fixed sampling times every t_s seconds, SAC measures the current (initial) state, x_{init}, and begins control synthesis by predicting the nominal motion of a robotic system over a receding horizon. Prediction starts at the current (initial) time, t_0, and extends to final time, t_f, with a horizon length, $T = t_f - t_0$. So, for example, SAC produces the 10 s SLIP trajectory in Fig. 1 by cycling through a synthesis process every $t_s = 0.01$ s (100 Hz), with each prediction phase lasting $T = 0.6$ s. In each cycle, SAC computes an action that improves the 0.6 s predicted trajectory. Repeating the process, SAC generates a piecewise continuous response.

In Part I, the dynamics,

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

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

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

The time dependence in (1) and (2) will be dropped for brevity.

The prediction phase simulates motion resulting from some choice of nominal control, $u = u_1$. Thus, the nominal predicted motion corresponds to,

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

Although the nominal control may be chosen arbitrarily, all examples here use a null nominal control, $u_1 = 0$. Hence, in the SLIP example, SAC seeks actions that improve performance relative to doing nothing, i.e., letting the SLIP fall.

With $l_1 : \mathbb{R}^n \mapsto \mathbb{R}$ and $m : \mathbb{R}^n \mapsto \mathbb{R}$, the cost functional,

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

measures trajectory performance to gauge the improvement provided by SAC actions.[3] The following assumptions further clarify the systems and cost functionals addressed.

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

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

The prediction phase concludes with simulation of (1) and (3).

B. Computing Optimal Actions

Since SAC has not yet decided *when* to act, it derives a schedule (curve), $u_2^* : (t_0, t_f) \mapsto \mathbb{R}^m$, providing the value of the optimal action at every moment along the predicted motion. For instance, in the SLIP example SAC may determine that 1 N-m of torque at the current time will tilt the leg backwards sufficiently for the SLIP to bounce forward and improve (3). The same strategy may be optimal at a later time, e.g., just before leg touchdown, but require 10 N-m to accelerate the leg into position before impact. In this scenario, u_2^* would provide these optimal torque values at each time and SAC would choose one "best" action to take. At every sample time, SAC would update u_2^* and choose another action.

Modeling a SAC action as a short perturbation in the predicted trajectory's nominal control, this section derives the optimal action to apply at a given time by finding the perturbation that optimizes trajectory improvement. Given the application time, $\tau \in (t_0, t_f)$, a (short) duration, λ, and the optimal action value, $u_2^*(\tau)$, the perturbed control signal is piecewise continuous based on Def. 1 and Assump. 3.

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

Definition 1. *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.*

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

$$u(t) = \begin{cases} u_1(t) & : t \notin [\tau - \frac{\lambda}{2}, \tau + \frac{\lambda}{2}] \\ u_2^*(\tau) & : t \in [\tau - \frac{\lambda}{2}, \tau + \frac{\lambda}{2}] \end{cases},$$

with nominal control, u_1, that is $\mathcal{C}^0$ in t.[4]

Hence, over each receding horizon, SAC assumes the system evolves according to nominal dynamics, f_1, except for a brief duration, where it switches to the alternate mode,

$$f_2 \triangleq f(x(t), u_2^*(\tau)).$$

SAC seeks the vector $u_2^*(\tau)$ that optimally improves cost (3).

In Part II, we derive a local model of the change in cost resulting from the perturbed SAC control signal and solve for actions that optimize improvement. In the present case of differentiable dynamics (1) however, the local model of the change in cost corresponds to an existing term from mode scheduling literature. That is, we can re-interpret the problem of finding the change in cost (3) due to short application of $u_2^*(\tau)$, as one of finding the change in cost due to inserting a new dynamic mode, f_2, into the nominal trajectory for a short duration around $t = \tau \in (t_0, t_f)$. In this case, the *mode insertion gradient* [9], [10],

$$\frac{dJ_1}{d\lambda^+}(\tau, u_2^*(\tau)) = \rho(\tau)^T \Big[f(x(\tau), u_2^*(\tau)) - f(x(\tau), u_1(\tau)) \Big], \tag{4}$$

provides a first-order model of the change in cost (3) relative to the duration of mode f_2. The model is local to the neighborhood where the duration of the switch to/from f_2 approaches zero, $\lambda \to 0^+$. Note that (4) assumes the state in f_1 and f_2 is defined from the nominal control and $\rho : \mathbb{R} \mapsto \mathbb{R}^n$ is the adjoint variable calculated from the nominal trajectory,[5]

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

with $\rho(t_f) = \nabla\, m(x(t_f))$.

The mode insertion gradient is typically used in mode scheduling [10], [13]–[15] to determine the optimal time to insert control modes assuming the modes are known a priori. In this section, we use the mode insertion gradient to solve for new optimal modes (optimal actions) at each instant.[6]

One way to interpret the mode insertion gradient is as a sensitivity. That is, the mode insertion gradient (4) indicates the sensitivity of cost (3) to an action's application duration at any potential application time, $\tau \in (t_0, t_f)$. To achieve a desired degree of cost improvement with each action, SAC uses a control objective to select optimal actions that drive the

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

[5]As opposed to traditional fixed-horizon optimal control methods [11], [12], this adjoint is easily computed because it does not depend on the closed-loop, optimal state $x^*(t, u_2^*(\tau))$.

[6]Also, Section V shows a local hybrid cost model yields a generalized version of (4) for hybrid impulsive dynamical systems with resets and objectives that depend on the control. A discussion is in Appendix B-B.

cost sensitivity (4) toward a desired negative value, $\alpha_d \in \mathbb{R}^-$. At any potential application time $\tau \in (t_0, t_f)$, the action value, $u_2(\tau)$, that minimizes,

$$l_2(\tau, u_2(\tau)) \triangleq \frac{1}{2}[\frac{dJ_1}{d\lambda^+}(\tau, u_2(\tau)) - \alpha_d]^2 + \frac{1}{2}\|u_2(\tau)\|_R^2, \quad (6)$$

minimizes control authority in achieving the desired sensitivity. The matrix $R = R^T > 0$ provides a metric on control effort. Because the space of positive semi-definite / definite cones is convex [16], (6) is convex with respect to action values, $u_2(\tau)$.

With Assumps. 1-3, the mode insertion gradient exists, is bounded, and (6) can be minimized with respect to $u_2(\tau)\ \forall \tau \in (t_0, t_f)$. The following theorem, which stems from early work in [17], finds this minimum to compute the schedule of optimal action values.

Theorem 1. *Define* $\Lambda \triangleq h(x)^T \rho \rho^T h(x)$. *The schedule providing the value of the optimal action,*

$$u_2^*(t) \triangleq \underset{u_2(t)}{\arg\min}\ l_2(t, u_2(t)) \qquad \forall t \in (t_0, t_f), \quad (7)$$

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

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

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

$$J_2 = \int_{t_0}^{t_f} l_2(t, u_2(t))\, dt\,. \quad (9)$$

Because the sum of convex functions is convex, and x in (4) depends only on 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,

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

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

A generalization of the Fundamental Lemma of Variational Calculus [18], implies $\frac{\partial l_2(\cdot,\cdot)}{\partial u_2} = 0$ at the optimizer. Solving

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

in terms of u_2^* confirms the optimal schedule is (8). ∎

To summarize, in computing optimal actions, SAC calculates a schedule providing the value of the optimal action at every possible application time along the predicted trajectory. These values optimize a local model of the change in cost relative to control duration at each application time. The model is provided by the mode insertion gradient (4). As a benefit of SAC, the schedule of optimal action values can be computed from a closed-form expression, (8), of the nominal state and adjoint (5) *even for non-convex tracking costs* (3).

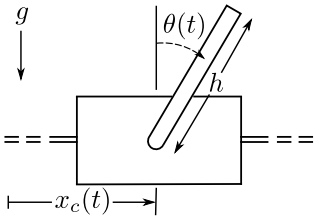


Fig. 4. Configuration variables for the cart-pendulum system.

C. Deciding When to Act

Assuming control calculations require some time, $t_{calc} < t_s$, SAC searches u_2^* for application times, $\tau \in (t_0 + t_{calc}, t_f)$,[7] that optimize an objective to find the most effective time to act over each predicted trajectory. We use

$$J_\tau(t) = \|u_2^*(t)\| + \frac{dJ_1}{d\lambda^+}(t, u_2^*(t)) + (t - t_0)^\beta\,, \quad (12)$$

to balance a trade-off between control efficiency and the cost of waiting, though there are many other choices of objective.[8]

Consider, for example, inverting a simple cart-pendulum with state, $x = (\theta,\, \dot\theta,\, x_c,\, \dot x_c)$ as in Fig. 4, acceleration control, $u = (a_c)$, and underactuated dynamics,

$$f(x,u) = \begin{pmatrix} \dot\theta \\ \frac{g}{h}\,\sin(\theta) + \frac{a_c\,\cos(\theta)}{h} \\ \dot x_c \\ a_c \end{pmatrix}, \quad (13)$$

with length, $h = 2\,\mathrm{m}$, and gravity, $g = 9.81\,\frac{\mathrm{m}}{\mathrm{s}^2}$. Fig. 5 shows a schedule, u_2^*, computed for (13) starting at $t_0 = 0.4\,s$ into an example closed-loop SAC trajectory. At every time, the action in u_2^* drives the mode insertion gradient (purple curve) toward $\alpha_d = -1,000$. The mode insertion gradient is 0 at $t \approx 1.39\,\mathrm{s}$ when the pendulum is horizontal, i.e., $\theta = \frac{\pi}{2}$ rad., since no action can push θ toward $\theta = 0$ at that time. The mode insertion gradient also goes to 0 toward the end of the horizon since no finite control action can improve (3) at the final time. The curve of J_τ vs time (blue) indicates, to optimize the trade-off between wait time and effectiveness of the action (as in (12)), SAC should do nothing until optimal time $t^* \approx 0.57\,\mathrm{s}$.[9]

D. Deciding How Long to Act

Temporal continuity of ρ, f_1, f_2 and u_1 provided by Assump. 1-3 ensures the mode insertion gradient is continuous with respect to duration around where $\lambda \to 0^+\ \forall\ \tau \in (t_0, t_f)$. Therefore, there exists a neighborhood, $V = \mathcal{N}(\lambda \to 0^+)$, where the sensitivity indicated by (4) models the change in cost relative to application duration to first-order (see [10], [13] and the generalized derivation in Section V). For finite durations, $\lambda \in V$, the change in cost (3) is locally modeled as

$$\Delta J_1 \approx \frac{dJ_1}{d\lambda^+}(\tau, u_2^*(\tau))\,\lambda\,. \quad (14)$$

[7]SAC implements the previous action while current calculations complete.

[8]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.

[9]The next SAC synthesis cycle, i.e., the next sampling time, may begin (and conclude) before the action at t^* is applied. In such cases, SAC often computes a similar t^*. So, in this case, SAC would likely continue to wait until $t^* \approx 0.57\,\mathrm{s}$ to act.

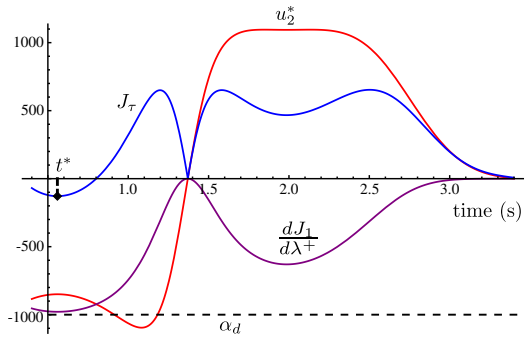


Fig. 5. A schedule of optimal actions, u_2^*, is depicted (red curve) for a $T = 3\,s$ predicted trajectory of the cart-pendulum system (13) starting at current time $t_0 = 0.4\,s$. These actions minimize acceleration in driving the mode insertion gradient toward $\alpha_d = -1,000$. The (purple) mode insertion gradient curve approximates the change in cost (3) achievable by short application of $u_2^*(t)$ at different times. The objective, J_τ, (blue curve) is minimized to find an optimal time, t^*, to act. Waiting to act at $\tau = t^*$ rather than at $\tau = t_0$ (we assume $t_{calc} = 0$), SAC generates greater cost reduction using less effort.

As $u_2^*(\tau)$ regulates $\frac{dJ_1}{d\lambda^+}(\tau, u_2^*(\tau)) \approx \alpha_d$, (14) becomes $\Delta J_1 \approx \alpha_d \lambda$. Thus the choice of λ and α_d allows the control designer to specify the desired degree of change provided by actions, $u_2^*(\tau)$. We use a line search with a simple descent condition to find a $\lambda \in V$ that yields the desired change [5].[10]

Upon selection of the application duration, λ, the SAC action is fully specified and sent to the robot. The process iterates and the next cycle begins when SAC incorporates new state feedback at the subsequent sampling time.

III. Special Properties of SAC Control

In addition to providing a closed-form solution for the entire schedule of optimal actions (8), SAC controls inherit powerful guarantees. Appendix A includes derivations that show 1) the schedule (8) globally optimizes (7). 2) Around equilibria, SAC controls simplify to linear state feedback laws permitting local stability analysis and parameter selection, e.g., parameters of (3), α_d, or T. 3) Finally, actions computed from (8) can be saturated to satisfy min-max constraints using quadratic programming, by scaling the control vector, or by scaling components of the control vector.[11]

For an overview of the SAC approach outlining the calculations required for on-line synthesis of constrained optimal actions, selection of actuation times, and resolution of control durations, refer to Algorithm 1.

IV. Example Systems

The following section provides simulation examples that apply SAC on-line in benchmark underactuated control tasks.[12] Each example emphasizes a different performance-related aspect of SAC and results are compared to alternative methods.

[10]Because the pair (α_d, λ) determines the change in cost each action can provide, it is worth noting that a sufficient decrease condition similar to the one proposed in [13] can be applied to the choice of λ.

[11]Proofs are included for each with $u_1 = 0$, as in all the examples in this paper. All examples enforce constraints using the component scaling approach.

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

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}, and the max backtracking iterations k_{max}.
while $t_{curr} < \infty$ **do**
  $(t_0, t_f) = (t_{curr}, t_{curr} + T)$
  Use feedback to initialize $x_{init} = x(t_0)$
  Simulate (x, ρ) from f_1 for $t \in [t_0, t_f]$
  Compute initial cost $J_{1,init}$
  Specify α_d [13]
  Compute u_2^* from (x, ρ) using Theorem 1
  Specify / search for time, $\tau > t_0 + t_{calc}$, to apply u_2^*
  Saturate $u_2^*(\tau)$
  Initialize $k = 0$, $J_{1,new} = \infty$
  while $J_{1,new} - J_{1,init} > \Delta J_{min}$ **and** $k \leq k_{max}$ **do**
    $\lambda = \omega^k \Delta t_{init}$
    $(\tau_0, \tau_f) = (\tau - \frac{\lambda}{2}, \tau + \frac{\lambda}{2})$
    Re-simulate x applying f_2 for $t \in [\tau_0, \tau_f]$
    Compute new cost $J_{1,new}$
    $k = k + 1$
  end while
  $u_1(t) = u_2^*(\tau)\ \forall t \in [\tau_0, \tau_f] \cap [t_0 + t_{calc}, t_0 + t_s + t_{calc}]$
  Send updated u_1 to robot
  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.

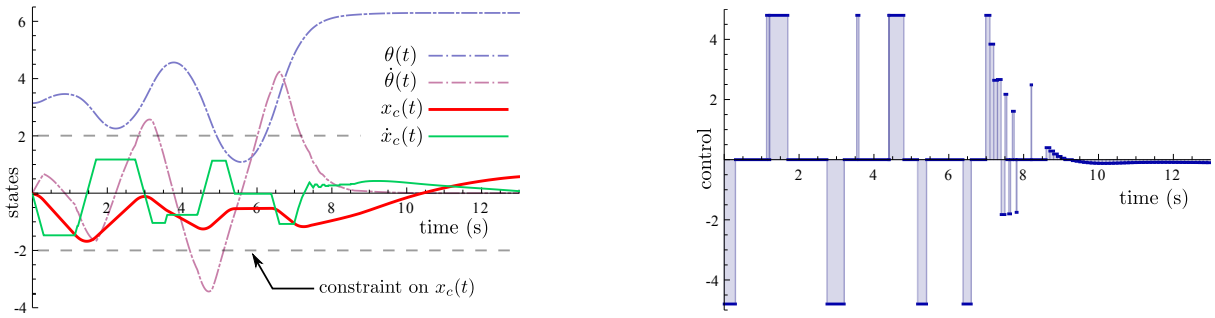
A. Cart-Pendulum

First, we present 3 examples where SAC is applied to the nonlinear cart-pendulum (13) in simulated constrained swing-up. Performance of SAC 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 [19]–[24]).

1) Low-Frequency Constrained Inversion: This example uses SAC to invert the cart-pendulum (13) 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 (31) with the state dependent weights, $Q(x(t)) = Diag[\,200, 0, (x_c(t)/2)^8, 50\,]$. to impose a barrier / penalty function (see [16], [25]) that constrains the cart's state so $x_c \in [-2, 2\,]$. Terminal and control costs in (6) and (31) are defined using $P_1 = 0$, $R = [0.3]$, and a horizon of $T = 1.5\,\text{s}$.[14]

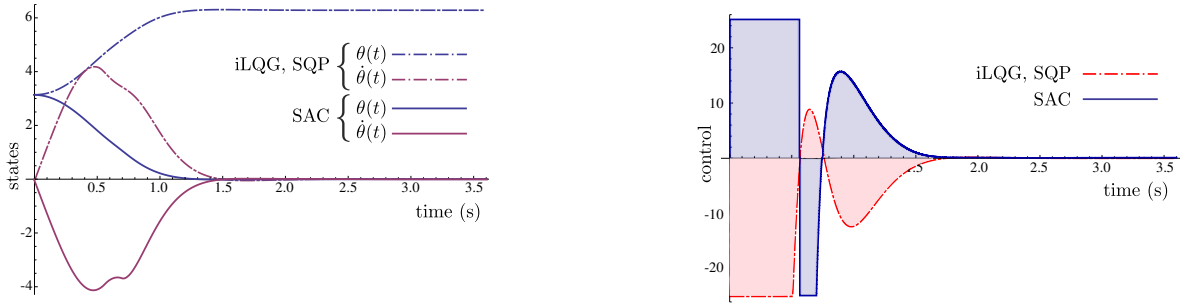
[13]In all examples, we choose to specify α_d as a feedback law, $\alpha_d = \gamma\, J_{1,init}, \gamma \in \mathbb{R}^-$. We find $\gamma \in [-15, -1]$ works well.

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



(a) (b)

Fig. 6. 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. 6b) illustrates how individual actions are sequenced (especially apparent from 7 to 10 s). SAC maintains the cart in $[-2, 2]$ m during inversion. Figure 6b also shows SAC automatically develops an energy pumping strategy to invert the pendulum.



(a) (b)

Fig. 7. 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.

Results in Fig. 6 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 $[-2, 2]$ m. The simulated trajectory is included in the video attachment.

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 off-line 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 [26]–[31]. The iLQG algorithm [32], [33] is a state-of-the-art variant of differential dynamic programming (DDP). While early versions did not accommodate control constraints, iLQG achieves a tenfold speed improvement over DDP in simulations [34] and has since been applied for real-time humanoid control [32]. This section compares to a recent variant that incorporates control constraints through a new active-set method [6]. We use a publicly available MATLAB iLQG implementation developed by its authors.[15]

To highlight the sensitivity of optimal control, i.e., iLQG and SQP, to local minima 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 (13). In this case, the goal is to compute controls that minimize a norm on the cart's acceleration while driving the pendulum angle toward the origin (inverted equilibrium). We compare performance of the trajectories produced by each algorithm over a fixed time horizon, T_{opt}, based on the objective,

$$J_{pend} = \frac{1}{2}\int_0^{T_{opt}} \|x(t) - x_d(t)\|_Q^2 + \|u(t)\|_R^2 \, dt \,, \qquad (15)$$

with $Q = Diag[1000,\ 10]$ and $R = [0.3]$. All algorithms are constrained to provide controls $\ddot{x}_c \leq |25| \, \frac{m}{s^2}$.

Both SQP and iLQG directly optimize a discretized version of (15) to derive their optimal trajectories. For comparison, results are provided for different choices of discretization, dt, and optimization horizons, T_{opt}.[16] In contrast, SAC computes a trajectory of duration T_{opt} by deriving actions from a receding state tracking cost (31) (see Appendix A-A) with quadratic state norms similar to the one in (15). Although SAC runs at

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

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

dt		$T_{opt} = 4\,\text{s}$		$T_{opt} = 5\,\text{s}$		$T_{opt} = 6\,\text{s}$	
		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 optimization horizon, T_{opt}, and discretization, dt. All solutions converge to the same optimizer with $J_{pend} \approx 2,215$, except the gray results, which converged to low performance local minima. For each parameter combination, columns indicate the number of iterations (iters) and time in minutes (min.) for convergence.

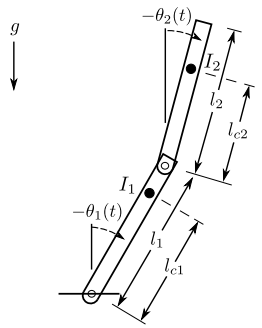


Fig. 8. Configuration of the acrobot and pendubot systems.

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.[17] Table II provides the time and number of optimization iterations required for each parameter combination.

The parameter combinations in Table II that do not correspond to gray data converged to the same (best case) optimal trajectory, which inverts the pendulum in $< 2\,$s with $J_{pend} \approx 2,215$. [18] Gray data indicate convergence to an alternate local minima with significantly worse cost. In all cases with $T_{opt} \neq 4\,$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 $T_{opt} = 6\,$s.

Since varying T_{opt} has no affect on SAC control synthesis (other than to specify the duration of the resulting trajectory), SAC control simulations included a variety of additional parameter combinations including receding horizons from $T = 0.15\,\text{s} - 3\,\text{s}$ and different synthesis frequencies. These solutions yield costs 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. 7 achieves the best case cost of $J_{pend} = 2,215$ from receding horizons of $T = 0.28\,$s, with parameters $Q = 0$ and $P_1 = Diag[500,\, 0]$ in (31), and with $R = [0.3]$. SAC's on-line controls 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 trajectory in Fig. 7 using 4 CPU cores. Our C++ implementation of SAC obtains a solution equivalent to the best results on-line, with 1 KHz feedback, in less than $\frac{1}{2}$ s using 1 CPU core.[19] 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 near the optimal solution (SAC provides feedback from arbitrary states), SQP or iLQG must be applied in receding horizon for feedback comparable to SAC. For improved speed, [6] recommends a receding horizon implementation using suboptimal solutions from a fixed number (one) of iterations. However, in this simple nonlinear example, SQP / iLQG trajectories only resemble the final solution a few iterations before convergence. Hence, receding horizon implementations would likely result in poor local solutions.

3) Sensitivity to Initial Conditions: Using a horizon of $T = 1.2\,$s, SAC was applied to invert the same, reduced cart-pendulum system from a variety of initial conditions. Simulations used the quadratic tracking cost (31) and weight matrices from (15). A total of 20 initial conditions for $\theta(t)$, uniformly sampled over $[0, 2\,\pi)\,$rad, were paired with initial angular velocities at 37 points uniformly sampled over $[0, 4\,\pi]\,\frac{\text{rad}}{\text{s}}$.

To gauge performance, a $10\,$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 $10\,$s from all initial conditions. The average computation time was $\approx 1\,$s for each $10\,$s trajectory on the test laptop.

B. Pendubot and Acrobot

This section applies SAC for swing-up control of the pendubot [35]–[37] and acrobot [38]–[40]. 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 [35] and experiments in [36]. The acrobot model parameters and dynamics are from simulations in [39] and in seminal work [38]. Figure 8 depicts

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

[18] The cost of the optimal solution is the same when measured for horizons $T_{opt} = 4 - 6\,$s since the incremental cost in (15) is negligible after inversion at $t \approx 2\,$s.

[19] 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 $5 - 35\,$s and are linear w.r.t. to horizon, T, and discretization, t_s.

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 [35], [37]–[42]). We also use LQR controllers to stabilize the systems once near the inverted equilibrium. However, the results here show SAC can swing-up both systems without special energy optimizing methods. The algorithm utilizes the quadratic state error based cost functional (31), without modification.

While the pendubot simulations in [35] require control torques up to a magnitude of $15\,\mathrm{N\,m}$ for inversion, the experimental results in [36] 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 [39].

Example simulations initialize each system at the downward, stable equilibrium and the desired position is the fully inverted equilibrium. Results are based on a feedback sampling rate of $200\,\mathrm{Hz}$ for the pendubot with $Q = Diag[100,\, 0.0001,\, 200,\, 0.0001]$, $P_1 = 0$, and $R = [0.1]$ and $400\,\mathrm{Hz}$ 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, respectively. We selected $|\theta_{1,2}| \leq 0.05$ as the switching condition for pendubot stabilization.[20] The acrobot switched once all its configuration variables were $\leq |0.25|$.

Figure 9 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. SAC inverts the pendubot using the same peak effort as in experiments from [36] and less than half that from simulations in [35]. Also, SAC requires only $3\,\mathrm{s}$ to invert, while simulations in [35] needed $\approx 4\,\mathrm{s}$. Where [35] switches 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 (31). 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 [39]). 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.

[20]More formally, a supervisory controller can switch between swing-up and stabilizing based on the stabilizing region of attraction [43], [44].

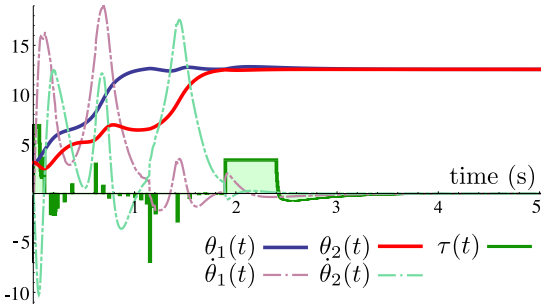


Fig. 9. SAC swings up the pendubot close enough for final stabilization by the LQR controller. The LQR controller takes effect at $t = 1.89$ s. The algorithm inverts the system using less peak control effort and in less time than existing methods from literature with the same parameters.

To invert the pendubot and acrobot in minimal time and under the tight input constraints, the two most important parameters for tuning are the horizon length, T, and the desired change in cost due to each control actuation, α_d. All examples specify α_d iteratively based on the current initial trajectory cost under the nominal (null) control as $\alpha_d = \gamma\, J_{1,init}$. Generally, because of the speed of SAC computations, good parameters 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 (31).

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 [32], [45], [46]. By planning each control action in a neighborhood of 0 duration, SAC avoids these issues and does not need to optimize control curves over discontinuous segments of trajectory.

V. Control Synthesis for Hybrid Systems

The SAC algorithm introduced in Section II is limited to differentiable nonlinear systems because the mode insertion gradient (4) is subject to Assump. 1. Rather than rely on (4), this section directly develops a first-order approximation of the variation in state and cost due to the perturbation in nominal

control generated by each SAC action. We show the change in cost due to short SAC actions corresponds to the same mode insertion gradient formula (4), but in terms of an adjoint variable derived 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 VI demonstrates the hybrid calculations on a $1D$ system and then illustrates SAC in simulated on-line control of a bouncing ball. The section concludes with the spring-loaded inverted pendulum (SLIP) example from the introduction.

A. Prediction

As in Part I, SAC predicts the nominal motion of hybrid robotic systems and computes actions that improve trajectory cost over (receding) horizons. However, this section introduces new notation more appropriate for hybrid impulsive systems. Specifically, the classes of hybrid systems considered here are similar to those in [46] and are defined such that:[21]

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*, $\Omega = \{\Omega_{q,q'} \in \mathcal{C}^1(\mathcal{M}_q, \mathcal{M}_{q'}) : (q,q') \in \mathcal{Q}\}$.

For clarity, we avoid using numerical subscripts for the nominal control, u_1. Instead, SAC predicts nominal motions assuming a (possibly null) nominal control, $u_{n,q} \in \mathcal{U}_q$, is defined for every location, $\forall q \in \mathcal{Q}$. So, for instance, in the SLIP example, one nominal control is defined in stance with another, possibly identical, control in flight. Note that as a hybrid robotic system applies controls, it evolves through an ordered sequence of locations, $(q_1, \ldots, q_r) : r \in \mathbb{N}$, e.g., from flight, to stance, to flight again, for a hopping SLIP.

With the initial location as q_1, state $x(t_0) = x_{init} \in \mathcal{M}_{q_1}$ and the collection $\{f, \Phi, \Omega\}$, SAC's prediction phase simulates

$$\dot{x}_{n,q_i} = f_{q_i}(x_{n,q_i}, u_{n,q_i}) : t \in \mathcal{I}_{q_i}, q_i \in \mathcal{Q}\,, \qquad (16)$$

starting with $i = 1$, to obtain the nominal state. Guards indicate when a transition should occur, i.e., they specify the end of each interval $\mathcal{I}_{q_i}$, and the next location, q_{i+1}, based on which guard becomes 0. Reset maps define the initial condition in q_{i+1} as $\{x_{n,q_{i+1}}(t_i^+) = \Omega_{q_i,q_{i+1}}(x_{n,q_i}(t_i^-)) : t_i^- \triangleq \sup \mathcal{I}_{q_i}, t_i^+ \triangleq \inf \mathcal{I}_{q_{i+1}}\}$. Through this process, the prediction phase defines the nominal location sequence, $(q_1, \ldots, q_r)$, intervals, $\mathcal{I}$, and the resulting nominal trajectory,

$$(x_n(t), u_n(t)) \triangleq (x_{n,q_i}(t), u_{n,q_i}(t)) : i \in \{1, \ldots, r\}, t \in \mathcal{I}_{q_i}\,.$$

As before, SAC's prediction phase concludes after computing the performance of the nominal trajectory. However, in this hybrid case we use an objective,

$$J = \int_{t_0}^{t_f} l(x(t), u(t))dt + m(x(t_f))\,, \qquad (17)$$

with incremental and terminal costs defined in each location, $\{l_{q_i} \in \mathcal{C}^1(\mathcal{M}_{q_i} \times U_{q_i}, \mathbb{R})\}_{q_i\in\mathcal{Q}}$ and $\{m_{q_i} \in \mathcal{C}^1(\mathcal{M}_{q_i}, \mathbb{R})\}_{q_i\in\mathcal{Q}}$, such that $l = l_{q_i} : t \in \mathcal{I}_{q_i}$ and $m = m_{q_i} : t \in \mathcal{I}_{q_i}$. Also, (17) is more general than (3), as it may depend on a control. Given $(x_n, \mathcal{I}, q_1, \ldots, q_r)$ resulting from nominal control, u_n, (17) can be evaluated along the hybrid trajectory.

B. Computing Optimal Actions

Recall that each cycle of SAC seeks an action that improves nominal trajectory performance. This section defines the perturbed signal from an arbitrary SAC action of value w as

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

assuming a short duration, $\lambda = \epsilon a$. In this case, 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 as $\lambda \to 0^+$, assume the perturbation occurs when the nominal state, x_n, is in the arbitrary location q_i so that $[\tau - \epsilon a, \tau] \subseteq \mathcal{I}_{q_i}$.[22] Figure 10 depicts the perturbed control and the corresponding perturbed state.

To derive actions that maximally improve the nominal trajectory, Sec. II-B used the mode insertion gradient (4) to model the change in cost (3) relative to control duration. To accommodate the discontinuous trajectories of hybrid robotic systems, this section derives a model of the change in nominal cost (17) resulting from the perturbed, u_w, by first modeling the effect of the control perturbation on state trajectory. To these ends, we define the first-order perturbed state model,[23]

$$x_w(t, \epsilon) \triangleq x_n(t) + \epsilon\Psi(t) + o(\epsilon)\,. \qquad (18)$$

The $\Psi(t)$ term is known as the *variational equation* [11], [12]. It is the direction of the state variation at time t and ϵ is the magnitude. The following proposition provides formulas to compute the variational equation along hybrid impulsive trajectories. The derivation is in Appendix B-A.

Proposition 1. *Assume the state, x_n, of a hybrid system evolves from location $q_i \in \mathcal{Q}$ to $q_{i+1} \in \mathcal{Q}$ with the transition*

[21] We assume actions are not applied at switching times, exclude Zeno behavior, and allow only a single element of Φ to be active 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 nominal location sequence (as in [46]).

[22] In the limit as $\lambda \to 0^+$, the SAC action is a needle perturbation [12].

[23] 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 (18) as $\epsilon \to 0$.

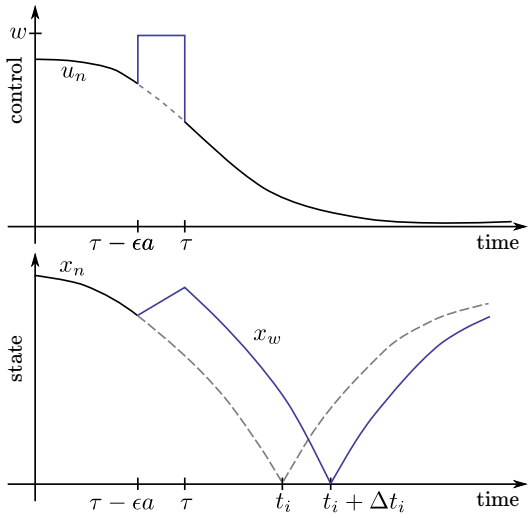


Fig. 10. A perturbed control (top) and the corresponding state variation (bottom) for a hybrid system. The nominal system switches locations at time t_i and the perturbed system switches at time $t_i + \Delta t$. Taken in the limit as $\epsilon a \to 0^+$, the control perturbation is a needle perturbation, which is equivalent to an infinitesimal duration action in SAC.

time, $t = t_i$. If a control perturbation occurs at $t = \tau < t_i$, as in Fig. 10, state variations propagate according to

$$\Psi \triangleq \begin{cases} \Big(f_{q_i}(x_n(\tau), w) - f_{q_i}(x_n(\tau), u_n(\tau))\Big) a & : t = \tau \in \mathcal{I}_{q_i} \\ \dot{\Psi} = A_{q_i} \Psi & : t \in (\tau, t_i^-] \\ \Psi(t_i^+) = \Pi_{q_i, q_{i+1}} \Psi(t_i^-) & : t = t_i^+ \\ \dot{\Psi} = A_{q_{i+1}} \Psi & : t \in (t_i^+, t_{i+1}^-] \end{cases}, \tag{19}$$

with the linear variational reset map,

$$\Pi_{q_i,q_{i+1}} \triangleq D_x \Omega_{q_i,q_{i+1}}(x_n(t_i^-)) \Big[I - f_{q_i}^- \frac{D_x \Phi_{q_i,q_{i+1}}(x_n(t_i^-))}{D_x \Phi_{q_i,q_{i+1}}(x_n(t_i^-)) f_{q_i}^-} \Big] + f_{q_{i+1}}^+ \frac{D_x \Phi_{q_i,q_{i+1}}(x_n(t_i^-))}{D_x \Phi_{q_i,q_{i+1}}(x_n(t_i^-)) f_{q_i}^-}, \tag{20}$$

$f_{q_i}(x_n(t_i^-), u_n(t_i^-)) \triangleq f_{q_i}^-$, $f_{q_{i+1}}(x_n(t_i^+), u_n(t_i^+)) \triangleq f_{q_i}^+$, and $A_{q_i}(t) \triangleq D_x f_{q_i}(x_n(t), u_n(t)) : t \in \mathcal{I}_{q_i}$ is the linearization about the (known) nominal state trajectory at q_i.

Note that if a nominal trajectory evolves through more than two locations, each transition requires reset of Ψ at transition times according to (20). Variations continue according to the dynamics linearized about the nominal trajectory. Repeating computations in rows $2 - 4$ of (19) between consecutive locations, variations can be propagated to $t = t_f$.

With Prop. 1 to compute the perturbed state (18), the following section derives the cost variation resulting from the perturbed control, u_w. We will show the formula is a generalization of the mode insertion gradient (4) that applies to a larger class of hybrid and impulsive systems.

1) Modeling the Cost Variation: To first-order, the perturbed cost can be modeled as,

$$J_w(x_n, u_n, \epsilon) \triangleq J|_{(x_n,u_n)} + \epsilon \nu(t_f) + o(\epsilon), \tag{21}$$

where $\nu(t_f)$ is the direction of variation in the cost function and ϵ is the magnitude. To simplify derivation of $\nu(t_f)$, we translate the hybrid system to Mayer form by appending the incremental costs, l_{q_i}, to the dynamics vectors, f_{q_i}, in each location. Objects with a bar refer to appended versions of hybrid system such that $\bar{f}_{q_i} = [\, l_{q_i},\, f_{q_i}^T \,]^T$ 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, the first component of the perturbed appended state, $\bar{x}_{w,1}(t_f, \epsilon)$, is the perturbed integral cost in (17). Hence, the perturbed cost model (21) can be written as a sum,

$$J_w(\bar{x}_w, \epsilon) \triangleq \bar{x}_{w,1}(t_f, \epsilon) + m(x_w(t_f, \epsilon)),$$

which includes the perturbed terminal cost. The direction of variation in the cost is $D_\epsilon J_w(\bar{x}_w, 0) \triangleq \nu(t_f) = \bar{\Psi}_1(t_f) + D_\epsilon m(x_w(t_f, 0))$. Evaluating the derivative yields

$$\begin{aligned} \nu(t_f) &= \bar{\Psi}_1(t_f) + D_x m(x(t_f)) \Psi(t_f) \\ &= [1, \nabla m(x(t_f))] \cdot \bar{\Psi}(t_f). \end{aligned} \tag{22}$$

Note that $\nu(t_f)$ provides the same information as the mode insertion gradient in (4) but applies to hybrid impulsive systems with resets. That is, $\nu(t_f)$ provides the sensitivity of a cost, J, to applying an action at $t = \tau$ as $\lambda \to 0^+$. Given a control perturbation at arbitrary time $\tau \in (t_0, t_f)$, one can calculate $\nu(t_f)$ from (22) by propagating the appended state variation forward from $t = \tau$ to $t = t_f$ using Prop. 1. However, in searching for an optimal time to act, SAC needs to compare the cost variation produced by taking action, i.e., applying a control perturbation, at different times, $\tau \in (t_0, t_f)$ (see Sec. II-C). The process is computationally intensive if $\nu(t_f)$ is naively computed from the state variation. That is, considering two possible times, $\tau < \tau'$, when control perturbation may be applied, $\nu(t_f)$ would require separate simulations of $\bar{\psi}$ from $[\tau, t_f]$ and $[\tau', t_f]$.[24]

Since the mode insertion gradient (4) does not require re-simulation to consider different application times $\tau \in (t_0, t_f)$ in optimizing (12), we seek to express $\nu(t_f)$ in a form that more closely resembles (4). To these ends, we now re-write $\nu(t_f)$ in terms of an *adjoint* system, $\bar{\rho}$,[25] to the variational system $\bar{\Psi}$.[26] The systems are adjoint [11] if

$$\frac{d}{dt}(\bar{\rho} \cdot \bar{\Psi}) = 0 = \dot{\bar{\rho}} \cdot \bar{\Psi} + \bar{\rho} \cdot \dot{\bar{\Psi}}. \tag{23}$$

That is, we can derive $\bar{\rho}$ by ensuring $\bar{\rho} \cdot \bar{\Psi}$ is constant. Note also that by choosing the terminal condition,

$$\bar{\rho}(t_f) = [\, 1,\, \nabla m(x(t_f))\,], \tag{24}$$

(22) allows us to express $\nu(t_f)$ in terms of the adjoint at the terminal time as $\nu(t_f) = \bar{\rho}(t_f) \cdot \bar{\Psi}(t_f)$. If we enforce (23) in deriving the adjoint, the inner product will be constant and

[24] 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.

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

[26] See [11] for a similar derivation of an adjoint in the context of continuous variations.

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 $\bar{\rho}(t)$ can be interpreted as the sensitivity of (17) to a state variation at time t.

Assuming the system is at the (arbitrary) location $q\in\mathcal{Q}$ at the perturbation time $t=\tau$, the inner product in (23) yields

$$\begin{aligned}\bar{\rho}(\tau)\cdot\bar{\Psi}(\tau) &= \bar{\rho}(\tau)\cdot\Big(\bar{f}_q(x_n(\tau),w)-\bar{f}_q(x_n(\tau),u_n(\tau))\Big)a\\ &= \nu(t_f)\,,\end{aligned}\tag{25}$$

which no longer depends on forward simulation of $\bar{\Psi}(\tau)$. The initial time, τ, of the control perturbation is arbitrary. Like in (4), once the adjoint, $\bar{\rho}$, is computed over $[t_0,t_f]$, (25) can be evaluated at any number of different times, $\tau\in(t_0,t_f)$, to provide the cost sensitivity, $\nu(t_f)$, to the control perturbation in each case.

The following proposition derives an adjoint formula, $\bar{\rho}$, that maintains its interpretation as the cost sensitivity to state variations, as in $\bar{\rho}(t)\cdot\bar{\Psi}(t)=\nu(t_f)\ \forall t\in[\tau,t_f]$.

Proposition 2. *Assuming x_n flows between the locations $q_i,q_{i+1}\in\mathcal{Q}$ with a control perturbation as in Prop. 1,*

$$\bar{\rho}\triangleq\begin{cases}[\,1,\,\nabla m(x(t_f))\,] & :t=t_f\\ \dot{\bar{\rho}}=-\bar{A}_{q_{i+1}}^T\bar{\rho} & :t\in[t_i^+,t_f)\\ \bar{\rho}(t_i^-)=\bar{\Pi}_{q_i,q_{i+1}}^T\bar{\rho}(t_i^+) & :t=t_i^-\\ \dot{\bar{\rho}}=-\bar{A}_{q_i}^T\bar{\rho} & :t\in[\tau,t_i^-)\end{cases},\tag{26}$$

satisfies the adjoint relation (23) *and* (25).

Proof: The adjoint is simulated backwards from a terminal condition (26) because this choice of terminal conditions yields $\nu(t_f)$ in (22). The continuous flow equations in rows 2 and 4 of (26) are the direct result of enforcing (23) with rows 2 and 4 of (19). Similarly, the reset equation results from application of the adjoint relation across the transition time,

$$\begin{aligned}\frac{d(\bar{\rho}(t_i)\cdot\bar{\Psi}(t_i))}{dt}=0&=\frac{\bar{\rho}(t_i^+)\cdot\bar{\Psi}(t_i^+)-\bar{\rho}(t_i^-)\cdot\bar{\Psi}(t_i^-)}{t_i^+-t_i^-}\\ 0&=\bar{\rho}(t_i^+)\cdot\bar{\Pi}_{q_i,q_{i+1}}\bar{\Psi}(t_i^-)-\bar{\rho}(t_i^-)\cdot\bar{\Psi}(t_i^-)\\ \bar{\rho}(t_i^-)&=\bar{\Pi}_{q_i,q_{i+1}}^T\bar{\rho}(t_i^+)\,.\end{aligned}$$

■

As for the variational equation, one may propagate $\bar{\rho}$ between arbitrary numbers of consecutive modes by repeating the reset and continuous flow steps in (26).

When the incremental costs, l, do not depend on the control, e.g., in (3), and w corresponds to the value of an optimal SAC action, (25) is equivalent to the mode insertion gradient (4).[27] Hence, the SAC process applies as-is to hybrid and impulsive systems. The user need only account for hybrid transitions in simulations, e.g., of state trajectory and the adjoint (26).

VI. Hybrid Control Examples

This section presents three illustrative examples using the hybrid methods just described. Section VI-A demonstrates calculation of the variational, adjoint, and *hybrid mode insertion gradient* (25) equations for a 1D example. Section VI-B uses the hybrid version of SAC (based on the adjoint in (26)) to control a bouncing ball through impacts and toward a goal state. Lastly, Sec. VI-C applies SAC to control a the hybrid spring-loaded inverted pendulum model up a flight of stairs.

[27] Appendix B-B details the connection between (4) and (25). The section also describes how (25) can consider dynamic modes that differ in more than control, to enable mode scheduling algorithms for more general classes of hybrid systems with resets.

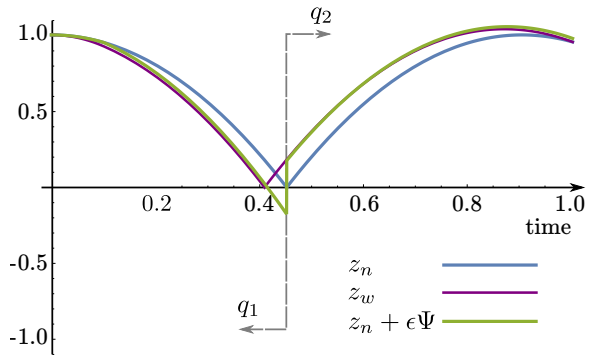


Fig. 11. The height $z(t)$ of a mass dropped from 1 m. The mass follows the nominal trajectory, z_n, and bounces at impact due to an elastic collision. The reset map reflects its velocity in transitioning from q_1 to q_2. The purple curve is the varied trajectory simulated from the hybrid impulsive dynamics with a variation at $\tau=0.1$ s of duration $\lambda=0.1$ s ($a=1$, $\epsilon=0.1$ s), in the nominal control. The variation accelerates the mass in the z direction at $w=-5\,\frac{\mathrm{m}}{\mathrm{s}^2}$. The green curve is the approximated trajectory based on the first-order model.

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 $z_0=1$ m with no initial velocity. It falls under gravity until it impacts with a flat surface (guard) at $z=0$ 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 $\dot{z}$ when the guard becomes 0 at impact. The simulation parameters follow.

System Parameters:

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

Control Perturbation:

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

Figure 11 shows the system's nominal trajectory (blue curve) and the varied trajectory resulting from a simulated control variation. 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 (19) are in Fig. 12. As Fig. 12 shows, the state variations are discontinuous at impact, while the direction of the cost variation, $\nu(t)$, is continuous over time. The dashed black line in Fig. 12 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

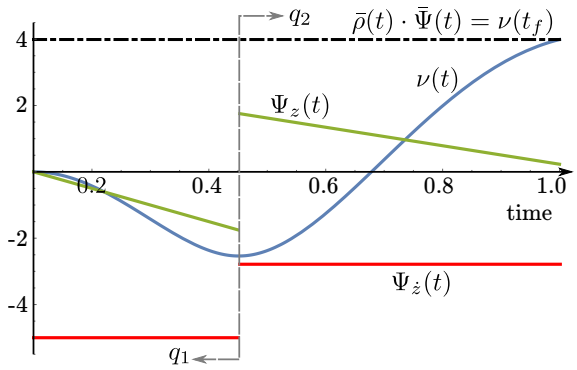


Fig. 12. The direction of state (and cost) variations, $\bar{\Psi} = (\, \nu,\, \Psi_z,\, \Psi_{\dot{z}} \,)$, resulting from a variation at $\tau = 0.1\,\text{s}$ of duration $\lambda = 0.1\,\text{s}$ ($a = 1$, $\epsilon = 0.1\,\text{s}$) in the nominal control. The control variation accelerates the 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.

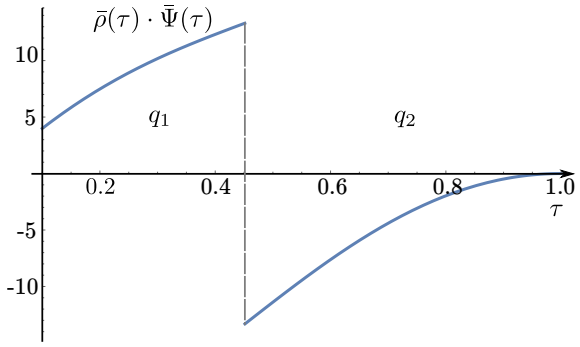


Fig. 13. The value of $\bar{\rho}(\tau) \cdot \bar{\Psi}(\tau)$ (according to (25)) versus τ. The term indicates the sensitivity of the performance objective to the control perturbation, $w = -5\,\frac{\text{m}}{\text{s}^2}$, if that perturbation were to occur at different points $\tau \in (t_0, t_f)$. Before impact, (25) indicates a short control perturbation will increase cost (17). After impact, a short control perturbation will lower cost.

perturbation, $\forall t \in [\tau, t_f]$. Figure 13 shows how this inner product (the value of (25)) would change if the control perturbation were applied at different times, $\tau \in (t_0, t_f)$.

Results:

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

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

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

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

As asserted earlier, the approximation of the change in cost (17) from (25) 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. 11 closely approximates the true perturbed trajectory, z_w, simulated from the perturbed control and the nonlinear dynamics. Additionally, (44) 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. 11. Figure 13 shows that (25) correctly indicates it will be helpful (reduce trajectory cost according to (17)) 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 (44) and (20) are well defined with $D_x\Phi_{q,q'}(x_n(t_i^-))f_{q_i}(x_n(t_i^-), u_n(t_i^-)) \neq 0 \;\forall (q, q') \in \mathcal{Q}$.

B. Control of A Bouncing Ball

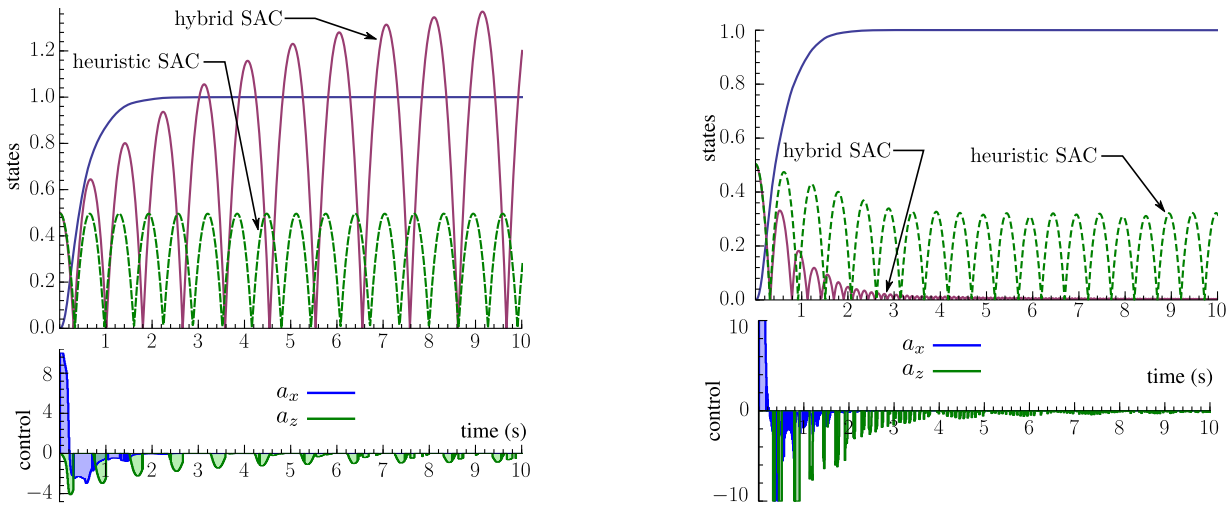
This section uses the SAC algorithm with the adjoint variable (26)[28] to develop closed-loop controls on-line that drive a hybrid impulsive bouncing ball model toward different desired states. The system state vector consists of the 2D position and velocity of the ball, $x = (\, x_b,\, z_b,\, \dot{x}_b,\, \dot{z}_b \,)$. The system inputs are constrained accelerations, $u = (\, a_x,\, a_z \,) : a_x \in [-10, 10]\,\frac{\text{m}}{\text{s}^2}$, $a_z \in [-10, 0]\,\frac{\text{m}}{\text{s}^2}$, and the dynamics are $f_q(x, u) = (\, \dot{x}_b,\, \dot{z}_b,\, a_x,\, a_z - g \,) : \forall q \in \mathcal{Q}$. 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 (31) 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 $100\,\text{Hz}$.[29] 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. 14a require $0.21\,\text{s}$ to simulate using the C++ SAC implementation from Section IV.

Accelerating the ball in the horizontal directions, SAC drives the ball toward the desired horizontal position $1\,\text{m}$ away. Due to the control constraints on a_z, however, SAC cannot achieve the desired height. Instead, Fig. 14a 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 (26) introduced here. Without the jump terms in the adjoint simulation (from reset map $\Pi_{q,q'}$), the mode insertion gradient (4) does not

[28]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)).

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



(a) (b)

Fig. 14. SAC accelerates a ball 1 m to the right and either up (Fig. 14a) or down (Fig. 14b). In both cases x_b (the blue state curve) reaches the desired point 1 m away. In Fig. 14a, 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. 14b, 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.

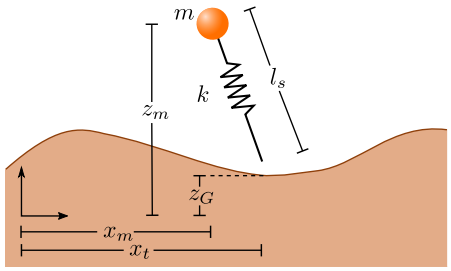


Fig. 15. Planar configuration variables for the SLIP.

switch signs at impact events as in Fig. 13 and so does not accurately model the sensitivity to control actions.

A similar demonstration in Fig. 14b 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 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 control, a_z. Lastly, if one applies the smooth version of SAC from Section II 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 m, it ceases to make further progress (see Fig. 14b). 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 SLIP model from the introduction. This section uses a 12 dimensional (9 states and 3 controls) model that is similar to the one in [47]. Figure 15 depicts the SLIP's planar configuration variables.

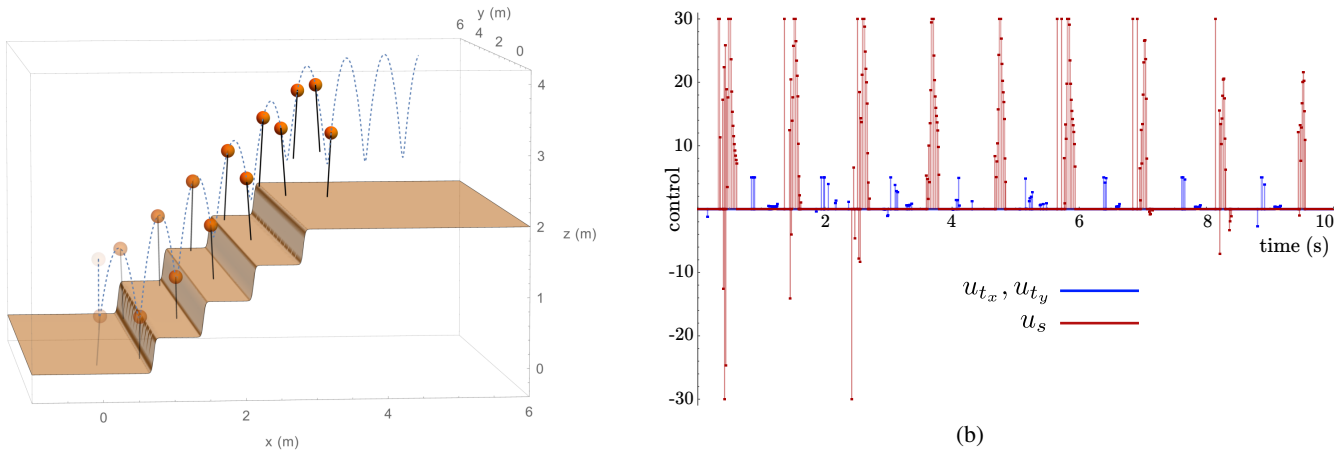
The SLIP's dynamics are divided into flight and stance modes. In our case, the state vector is the same for each mode and includes the 3D position / velocity of the mass, the 2D position of the spring endpoint ("toe"), and a book-keeping variable, $q \in \{\text{f}, \text{s}\}$, tracking the current 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, $u = (u_{t_x}, u_{t_y}, u_s)$, 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,

$$f_{\text{s}}(x,u) = \begin{pmatrix} \dot{x}_m \\ \frac{(k(l_0-l_s)+u_s)(x_m-x_t)}{ml_s} \\ \dot{y}_m \\ \frac{(k(l_0-l_s)+u_s)(y_m-y_t)}{ml_s} \\ \dot{z}_m \\ \frac{(k(l_0-l_s)+u_s)(z_m-z_G)}{ml_s} - g \\ 0 \\ 0 \end{pmatrix}, \tag{27}$$

define the first hybrid mode, and flight dynamics,

$$f_{\text{f}}(x,u) = \big(\dot{x}_m,\ 0,\ \dot{y}_m,\ 0,\ \dot{z}_m,\ -g,\ \dot{x}_m + u_{t_x},\ \dot{y}_m + u_{t_y}\big), \tag{28}$$



(a) (b)

Fig. 16. A time lapse showing the SLIP at 0.5 s increments (Fig. 16a) under SAC controls (Fig. 16b).

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,

$$l_s = \sqrt{(x_m - x_t)^2 + (y_m - y_t)^2 + (z_m - z_G)^2}. \tag{29}$$

When $l_s = l_0$, the guard equations,

$$\Phi_{\text{f,s}}(x) = \Phi_{\text{s,f}}(x) = z_m - \frac{l_0(z_m - z_G)}{l_s} - z_G\,, \tag{30}$$

cross zero to indicate 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 $\Omega_{\text{f,s}}$ and $\Omega_{\text{s,f}}$ leave the state unchanged other than to update the location variable, q.

Figure 16 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, $z_G = \sum_{n=1}^{4} \frac{0.5}{1 + e^{-75(x - 0.7n)}}$. 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.[30] 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 [1]. 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 for stable hopping that control the SLIP leg to desired touchdown angles. These methods typically assume the leg can swing arbitrarily fast to implement analytically computed touchdown angles, ignoring 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 [48]–[50]. 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.

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

VII. 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 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 [32], [51].

To better automate policy generation and reduce required user input, SAC needs tools for parameter tuning, especially ones that provide stability. As mentioned in Appendix A-A, SAC parameters can be selected to provide local stability around equilibrium based on a linear state feedback law for optimal actions (32). Sums-of-Squares (SOS) tools, e.g., the S-procedure [52], [53], 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 [51], [54]–[57]. 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 [58]–[62]. It offers a reactive, on-line control process that can reduce complexity and pre-computation required for robotic perching and aviation experiments in [53], [63], [64]. Its speed may facilitate predictive feedback control for new flexible robots [65], [66] and systems that are currently restricted to open-loop. It offers real-time system ID and parameter estimation for nonlinear systems [67]–[69]. These potential applications merit study and further development of the SAC approach.

APPENDIX A

The following sections highlight useful properties of SAC controls resulting from the synthesis process in Sec. II.

A. SAC Control Guarantees

Recall that SAC derives a schedule, u_2^*, that minimizes a convex objective (9), and causes the continuous first variation (10) to vanish locally. These are necessary and sufficient for the results in Corollary 1.

Corollary 1. *Solutions u_2^*, in* (8)*, exist, are unique, and globally optimize the control cost, J_2, in* (9)*.*

Additionally, the following Corollary 2 shows that 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) *is time invariant with an equilibrium, $(x, u) = 0$, the state tracking cost* (3) *is quadratic,*[31]

$$J_1 = \frac{1}{2}\int_{t_0}^{t_f} \|x(t)-x_d(t)\|_Q^2\, dt + \frac{1}{2}\,\|x(t_f)-x_d(t_f)\|_{P_1}^2\,, \quad (31)$$

with $x_d = x_d(t_f) = 0$, $Q = Q^T \geq 0$, and $P_1 = {P_1}^T \geq 0$, and $u_1 = 0$. There exists a neighborhood, $\mathcal{N}(x = 0)$, where optimal actions (8) *are linear feedback regulators,*

$$u_2^*(t) = \alpha_d\, R^{-1} h(0)^T\, P(t)\, x(t) \quad \forall t \in (t_0, t_f)\,. \quad (32)$$

[31]Quadratic cost (31) is assumed so that resulting equations emphasize the local similarity between SAC controls and LQR [70].

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

$$\rho(t) = P(t)\, x(t) \quad \forall t \in \mathcal{N}(t = t_f)\,. \quad (33)$$

Applying this relationship, (8) can formulated as

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

This expression contains terms quadratic in x. For $x \in \mathcal{N}(x = 0)$, these quadratic terms go to zero faster than the linear terms, and controls converge to (32).

Near the equilibrium, the dynamics can be approximated to first order as $\dot{x} \approx A\, x + B\, u$, with LTI linearizations $A = D_x f(0, 0)$ and $B = D_u f(0, 0)$. Note the state in (33) is the nominal state from control, $u = u_1$, and is assumed in $\mathcal{N}(x = 0)$. Thus, when $u_1 \in \mathcal{N}(u = 0)$, the system is near the equilibrium and (33) can be differentiated using the approximated dynamics (with $u = u_1$) and (5) to show

$$\dot{\rho} = \dot{P}\, x + P\, \dot{x} \quad (34)$$
$$-Qx - A^T P\, x = \dot{P}\, x + P\, (A\, x + B\, u_1)\,.$$

When $u_1 = 0$, (34) reduces to

$$0 = Q + A^T P + P\, A + \dot{P}\,. \quad (35)$$

Note the similarity to a Lyapunov equation. Though we have only proved this relationship exists in neighborhoods $\mathcal{N}(t = t_f)$ and $\mathcal{N}(x = 0)$, because (35) is linear in P, (35) cannot exhibit finite escape time. Through a global version of the Picard–Lindelöf theorem [71], it is straightforward to verify (35) (and (32)) exists and is unique for arbitrary horizons and not only for $t \in \mathcal{N}(t = t_f)$. Hence, one can compute the time varying linear feedback regulators (32)[32] for $x \in \mathcal{N}(x = 0)$ with $P(t)$ from (35) and $P(t_f) = P_1$. ■

Assuming time invariant dynamics, a fixed horizon, T, and SAC continuously applies actions at the (receding) initial time, $t = t_0$, (32) yields a *constant* feedback law, $u_2^*(t) = -K\, x(t)$, where K depends on the linearizations, weights, Q, R, and P_1, the time horizon, T, and the α_d term. Thus LTI stability conditions may be applied to facilitate parameter selection.[33] Similarly, one can also show Corollary 2 yields a feedback expression in error coordinates for which LTV stability analysis can be used to identify parameters that guarantee local stability to a desired trajectory, $x_d(t)$.[34]

As a final point, if (3) is quadratic and the nominal control, u_1, modeled as applying consecutively computed optimal actions (32) near equilibrium, (35) becomes a Riccati differential equation for the closed-loop system (see [72]) and actions (32)

[32]Note the $h(0)^T = B^T$ term in (32) shows up because the system is assumed to be in a neighborhood where the dynamics can be linearly modeled.

[33]As an example, Sums-of-Squares (SOS) [52], [53] techniques can pre-compute regions of attraction for (32). These SOS methods can be applied for parameter optimization, or to determine when SAC should switch to continuous application of (32).

[34]In the LTV case, one would need to pre-compute the feedback matrix, $K(t)$, from each sample time when tracking a desired trajectory, (x_d, u_d), in simulation. Assuming fixing application times, e.g., $\tau = t_0$, for each action, one could interpolate between the constant feedback matrices to develop a single LTV feedback law for analysis, e.g., using SOS techniques [52], [53].

simplify to finite horizon LQR controls [70]. In this case one can prove the existence of a Lyapunov function ((35) with $\dot{P} = 0$) and guarantee stability for SAC using methods from LQR theory [72] 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 [4], [73]–[77].

B. 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, we assume $u_1 = 0$, 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 (36) subject to inequality constraints. The following proposition provides the resulting quadratic programming problem in the case of $u_1 = 0$.

Proposition 3. *At any application time τ, a control action exists that obeys saturation constraints from the constrained quadratic programming problem*

$$u_2^*(\tau) = \underset{u_2(\tau)}{\arg\min} \ \frac{1}{2}\|\Gamma(\tau)\, u_2(\tau) - \alpha_d\|^2 + \frac{1}{2}\|u_2(\tau)\|_R^2 \quad (36)$$

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

Proof: For control-affine systems with $u_1 = 0$, the mode insertion gradient (4) simplifies to the inner product,

$$\frac{dJ_1}{d\lambda^+}(\tau, u_2^*(\tau)) = \langle\, \Gamma(\tau)^T, u_2^*(\tau)\,\rangle\,. \quad (37)$$

With the linear mode insertion gradient (37), minimizing (36) subject to $u_{min,k} \leq u_{2,k}^*(\tau) \leq u_{max,k}\ \forall k$ is equivalent to optimizing (6) at time τ to find a saturated action, $u_2^*(\tau)$. ∎

Prop. 3 considers a constrained optimal action, $u_2^*(\tau)$, 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.[35] The linear relationship between $u_2^*(\tau)$ and α_d implies that if any component $u_{2,k}^*(\tau) > u_{max,k}$ or $u_{2,k}^*(\tau) < u_{min,k}$, 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 $u_{2,k}^*(\tau) > u_{max,k}$, choosing $\hat{\alpha}_d = \alpha_d\, u_{max,k}/u_{2,k}^*(\tau)$ will produce a positively scaled $u_2^*(\tau)$ that obeys all constraints. Linearity between $u_2^*(\tau)$ 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 4. *For the choice $\alpha_d < 0$, a control action $u_2^*(\tau)$ evaluated anywhere that $\Gamma(\tau)^T \triangleq h(x(\tau))^T\, \rho(\tau) \neq 0 \in \mathbb{R}^m$ will result in a negative mode insertion gradient* (4) *and so can reduce* (3)*.*

Proof: Combining (8) with (37), optimal actions that reduce cost result in a mode insertion gradient satisfying

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

The outer product, $\Gamma(\tau)^T\Gamma(\tau)$, 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 $\|\Gamma(\tau)^T\|^2_{(\Gamma(\tau)^T\Gamma(\tau)+R^T)^{-1}} > 0$ for $\Gamma(\tau)^T \neq 0 \in \mathbb{R}^m$. Therefore, only choices $\alpha_d < 0$ in (9) produce optimal control actions that make $\frac{dJ_1}{d\lambda^+} < 0$ and by (14) 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, $u_2^*(\tau)$, 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 5. *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, τ, where $\Gamma(\tau)^T \triangleq h(x(\tau))^T\, \rho(\tau) \neq 0 \in \mathbb{R}^m$ can be independently saturated. If $\|u_2^*(\tau)\| \neq 0$ after saturation, the action is guaranteed to be capable of reducing cost* (3)*.*

Proof: For the assumptions stated in Prop. 5,

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

The outer product, $\Gamma(\tau)^T\Gamma(\tau)$, produces a rank 1 positive semi-definite, symmetric matrix with non-zero eigenvalue $= \Gamma(\tau)\Gamma(\tau)^T$ associated with eigenvector $\Gamma(\tau)^T$. Eigenvalue decomposition of the outer product yields $\Gamma(\tau)^T\Gamma(\tau) = S\, D\, S^{-1}$, where the columns of S corresponds to the eigenvec-

[35] Generally, scaling α_d will not equivalently scale the overall change in cost because the neighborhood, V, where the (14) models the change in cost can change. This would result in a different duration λ for the scaled action.

tors of $\Gamma(\tau)^T\Gamma(\tau)$ and D is a diagonal matrix of eigenvalues. For $R = R^T = c\,I$, actions satisfy

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

The matrix $D + c\,I$ 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, $\Gamma(\tau)\Gamma(\tau)^T + c$, applies to eigenvectors that are scalar multiples of $\Gamma(\tau)^T$. After inversion, $S\,(D + c\,I)^{-1}\,S^{-1}$ must then have an eigenvalue $\frac{1}{\Gamma(\tau)\Gamma(\tau)^T + c}$. Since inversion of a diagonal matrix leaves its eigenvectors unchanged, the eigenvalue scales $\Gamma(\tau)^T$. Therefore, the matrix $S\,(D + c\,I)^{-1}\,S^{-1}$ directly scales its eigenvector, $\Gamma(\tau)^T$, and

$$u_2^*(\tau) = \frac{\alpha_d}{\Gamma(\tau)\Gamma(\tau)^T + c}\,\Gamma(\tau)^T\,. \tag{38}$$

For any $\alpha_d \in \mathbb{R}^-$, $u_2^*(\tau)$ will be a negative scalar multiple of $\Gamma(\tau)^T$. 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 $u_2^*(\tau)$ and $\Gamma(\tau)^T$. The Law of Cosines verifies that control (38) is $180°$ relative to $\Gamma(\tau)^T$. Therefore, (38) corresponds to the control of least Euclidean norm that minimizes (37) and so maximizes the expected change in cost. The Law of Cosines and (37) also imply the existence of a hyperplane, $h_p := \{\nu(\tau) \in \mathbb{R}^m \,|\, \langle\, \Gamma(\tau)^T, \nu(\tau)\,\rangle = 0\}$, of control actions, $\nu(\tau)$, orthogonal to both (38) and $\Gamma(\tau)^T$. 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 (37)) and those that cannot.

To show that saturation returns a vector in the same subspace as (38), 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 $u_2^*(\tau) = a\,\hat{e}$. The Law of Cosines confirms that $u_2^*(\tau)$ 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 $\langle\, \hat{e}_k, u_2^*(\tau)\,\rangle = 0$, implying $a_k = 0$. Similarly, an a_k cannot be associated with an $\hat{e}_k > 90°$ relative to the control because this is equivalent to $\langle\, \hat{e}_k, u_2^*(\tau)\,\rangle < 0$, and leads to an $a_k < 0$ that contradicts definition.

Because (38) is represented by positively scaled bases within $90°$ of $u_2^*(\tau)$, all these vectors must lie on the same side of h_p as (38). This is also true of any vector produced by a non-negative linear combination of the components of $u_2^*(\tau)$. Since there always exists factors $\in [0, \infty)$, that can scale the elements of an action vector until they obey constraints $u_{min,k} \le 0 \le u_{max,k}\ \forall k \in \{1, \ldots, m\}$, saturated versions of (38) will still be capable of reducing cost for $\|u_2(\tau)\| \neq 0$. ■

APPENDIX B

The following appendix sections apply to the hybrid version of SAC in Part II. Specifically, Appendix B-A derives the formula for state variations in Prop. 1. Appendix B-B describes how (25) generalizes the mode insertion gradient and applies to mode scheduling problems for hybrid impulsive systems.

A. Computing the Varied State

According to Prop. 1, assume a control perturbation occurs at location $q_i \in \mathcal{Q}$ at $t = \tau$. While the system remains in the same location (no hybrid transitions), [11] shows the direction of state variations along the continuous trajectory segment satisfies,

$$\dot{\Psi} = A_{q_i}\Psi : t \in \mathcal{I}_{q_i}\,, \tag{39}$$

$$\Psi(\tau) = \Big(f_{q_i}(x_n(\tau), w) - f_{q_i}(x_n(\tau), u_n(\tau))\Big)a\,, \tag{40}$$

with $A_{q_i}(t) \triangleq D_x f_{q_i}(x_n(t), u_n(t)) : t \in \mathcal{I}_{q_i}$.

To propagate the varied state (18) to a new location q_{i+1}, we apply the reset map, $\Omega_{q_i,q_{i+1}}$, as in

$$\begin{aligned} x_w(t,\epsilon) = \Omega_{q_i,q_{i+1}}\bigg(x_w(t_i,\epsilon) + \int_{t_i}^{t_i+\Delta t_i^-} f_{q_i}(x_w(s,\epsilon), u_n(s))ds\bigg) \\ + \int_{t_i+\Delta t_i^+}^{t} f_{q_{i+1}}(x_w(s,\epsilon), u_n(s))ds \quad : t \in \mathcal{I}_{q+1}\,. \end{aligned} \tag{41}$$

Note the nominal state, x_n, transitions from q_i to q_{i+1} at t_i, $\Delta t_i \triangleq \Delta t_i(\epsilon)$ is the change in transition time due to the state variation (see Fig. 10), and a "$-$" or "$+$" superscript (as in $t_i + \Delta t_i^+$) indicates the time just before or after the transition.

To obtain the first-order variational equation for Ψ at q_{i+1} due to a control perturbation at q_i, we differentiate (41) as $\epsilon \to 0$ with $f_{q_i}(x_n(t_i^-), u_n(t_i^-)) \triangleq f_{q_i}^-$ and $f_{q_{i+1}}(x_n(t_i^+), u_n(t_i^+)) \triangleq f_{q_i}^+$, as in

$$\begin{aligned} \Psi(t) = D_x\Omega_{q_i,q_{i+1}}(x_n(t_i^-))\left[\Psi(t_i^-) + \frac{d\Delta t_i}{d\epsilon}\bigg|_{\epsilon\to 0} f_{q_i}^-\right] \\ - \frac{d\Delta t_i}{d\epsilon}\bigg|_{\epsilon\to 0} f_{q_{i+1}}^+ + \int_{t_i^+}^{t} A_{q_{i+1}}(s)\Psi(s)ds \quad : t \in \mathcal{I}_{q_{i+1}}. \end{aligned} \tag{42}$$

We compute $\frac{d\Delta t_i}{d\epsilon}\big|_{\epsilon\to 0}$ by locally enforcing the guard equation,

$$\Phi_{q_i,q_{i+1}}(x_w(t_i + \Delta t_i^-, \epsilon)) = 0\,, \tag{43}$$

using the first-order Taylor expansion of (43) around $\epsilon \to 0$. Applying $\Phi_{q_i,q_{i+1}}(x_n(t_i^-)) = 0$ in the expansion yields,

$$\frac{d\Delta t_i}{d\epsilon}\bigg|_{\epsilon\to 0} = -\frac{D_x\Phi_{q_i,q_{i+1}}(x_n(t_i^-))\Psi(t_i^-)}{D_x\Phi_{q_i,q_{i+1}}(x_n(t_i^-))f_{q_i}^-}\,. \tag{44}$$

Finally, one can define the new reset term, $\Pi_{q_i,q_{i+1}}$, according to (20), and use (39), (40) and (44) to express (42) as,

$$\Psi(t) = \Pi_{q_i,q_{i+1}}\Psi(t_i^-) + \int_{t_i^+}^{t} A_{q_{i+1}}(s)\Psi(s)ds \quad : t \in \mathcal{I}_{q_{i+1}}\,.$$

Just as $\Omega_{q_i,q_{i+1}}$ resets the state in (41), $\Pi_{q_i,q_{i+1}}$ provides a (linear) reset map that transitions Ψ between locations in (45).

Rather than calculate from (45), one can compute Ψ from the series of differential equations and resets in Prop. 1.

B. The Hybrid Mode Insertion Gradient

Part I of this paper makes use of the mode insertion gradient to locally model the changes in cost (3) generated by SAC actions. As in Sec. II, the mode insertion gradient in hybrid systems literature [10], [13]–[15], [78] applies to incremental costs, l_1, that do not depend on the control. With this assumption, the mode insertion gradient provides the sensitivity of J_1 to insertion of a new dynamic mode, i.e., switching from some nominal mode, $f_{q\in\mathcal{Q}}$, to another mode of the same dimension, $f_{q'\in\mathcal{Q}}$, for a short duration around $\lambda \to 0^+$. In the case of SAC, the alternate dynamic modes differ only in control and so result in the form of the mode insertion gradient in (4) for smooth systems. The expression (25) is a generalization of (4) that applies to hybrid impulsive systems and to costs (17), which may depend on the control.

Using the methods presented, it is straightforward to modify the initial condition of the variational equation (40) to accommodate an arbitrary dynamic mode insertion, $f_{q'}$, rather than a control perturbation. Note the formulas for the variational flow (19) and its corresponding adjoint equation would remain unchanged. In this case, (25) 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. Hence, the derivations and hybrid adjoint and mode insertion gradient calculations (25) introduced in Part II can enable mode scheduling algorithms like those in [13]–[15], [78] for these larger classes of hybrid impulsive systems.

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] H. Geyer, A. Seyfarth, and R. Blickhan, "Compliant leg behaviour explains basic dynamics of walking and running," *Proc. of the Roy. Soc. B: Biological Sciences*, vol. 273, no. 1603, pp. 2861–2867, Nov. 2006.

[2] M. Srinivasan and A. Ruina, "Computer optimization of a minimal biped model discovers walking and running," *Nature*, vol. 439, no. 7072, pp. 72–75, Jan. 2006.

[3] F. Allgöwer and A. Zheng, *Nonlinear model predictive control*. Basel, Switzerland: Birkhäuser Basel, 2000, vol. 26.

[4] F. Allgower, R. Findeisen, and Z. K. Nagy, "Nonlinear model predictive control: From theory to application," *J. Chinese Inst. of Chemical Engineers*, vol. 35, no. 3, pp. 299–316, May 2004.

[5] S. J. Wright and J. Nocedal, *Numerical optimization*, 2nd ed. New York, NY, USA: Springer Science + Business Media, 2006.

[6] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *Proc. IEEE Conf. Robotics and Automation*. IEEE, 2014, pp. 1168–1175.

[7] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, vol. 3, May 2009, p. 5.

[8] E. R. Johnson and T. D. Murphey, "Scalable Variational Integrators for Constrained Mechanical Systems in Generalized Coordinates," *IEEE Trans. Robot.*, vol. 25, no. 6, pp. 1249–1261, Dec. 2009.

[9] M. Egerstedt, Y. Wardi, and H. Axelsson, "Optimal control of switching times in hybrid systems," in *Int. Conf. on Methods and Models in Automation and Robotics*, 2003.

[10] ——, "Transition-time optimization for switched-mode dynamical systems," *IEEE Trans. Autom. Control*, vol. 51, no. 1, pp. 110–115, Jan. 2006.

[11] D. Liberzon, *Calculus of variations and optimal control theory: a concise introduction*. Princeton, New Jersey, USA: Princeton University Press, 2012.

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

[13] T. M. Caldwell and T. D. Murphey, "Projection-based optimal mode scheduling," *Nonlinear Anal.: Hybrid Syst.*, vol. 21, pp. 59–83, Dec. 2016.

[14] Y. Wardi, M. Egerstedt, and P. Twu, "A controlled-precision algorithm for mode-switching optimization," in *Proc. IEEE Conf. Decision and Control*, 2012, pp. 713–718.

[15] Y. Wardi and M. Egerstedt, "Algorithm for optimal mode scheduling in switched systems," in *Proc. Amer. Control Conf.*, 2012, pp. 4546–4551.

[16] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge, UK: Cambridge University Press, 2004.

[17] A. R. Ansari and T. D. Murphey, "Control-on-request: Short-burst assistive control for long time horizon improvement," in *Proc. Amer. Control Conf.*, 2015, pp. 1173–1180.

[18] D. S. Naidu, *Optimal control systems*. Boca Raton, FL, USA: CRC Press, 2003.

[19] K. J. Åström and K. Furuta, "Swinging up a pendulum by energy control," *Automatica*, vol. 36, no. 2, pp. 287–295, Feb. 2000.

[20] A. M. Bloch, M. Leok, J. E. Marsden, and D. V. Zenkov, "Controlled Lagrangians and stabilization of the discrete cart-pendulum system," in *Proc. IEEE Conf. Decision and Control and European Control Conf.*, 2005, pp. 6579–6584.

[21] R. Lozano, I. Fantoni, and D. J. Block, "Stabilization of the inverted pendulum around its homoclinic orbit," *Syst. & Control Lett.*, vol. 40, no. 3, pp. 197–204, Jul. 2000.

[22] N. Muskinja and B. Tovornik, "Swinging up and stabilization of a real inverted pendulum," *IEEE Trans. Ind. Electron.*, vol. 53, no. 2, pp. 631–639, Apr. 2006.

[23] B. Srinivasan, P. Huguenin, and D. Bonvin, "Global stabilization of an inverted pendulum–control strategy and experimental verification," *Automatica*, vol. 45, no. 1, pp. 265–269, Jan. 2009.

[24] J.-H. Yang, S.-Y. Shim, J.-H. Seo, and Y.-S. Lee, "Swing-up control for an inverted pendulum with restricted cart rail length," *Int. J. of Control, Automation and Syst.*, vol. 7, no. 4, pp. 674–680, Aug. 2009.

[25] R. Vinter, *Optimal control*. New York, NY, USA: Springer, 2010.

[26] B. C. Fabien, "Implementation of a robust SQP algorithm," *Optimization Methods & Software*, vol. 23, no. 6, pp. 827–846, Dec. 2008.

[27] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM Journal on Optimization*, vol. 12, no. 4, pp. 979–1006, Apr. 2002.

[28] T. Johansen, T. Fossen, and S. Berge, "Constrained nonlinear control allocation with singularity avoidance using sequential quadratic programming," *IEEE Trans. Control Syst. Technol.*, vol. 12, no. 1, pp. 211–216, Jan. 2004.

[29] S. Leyffer and A. Mahajan, *Software for nonlinearly constrained optimization*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2010.

[30] K. Schittkowski, "NLPQLP: A Fortran implementation of a sequential quadratic programming algorithm with distributed and non-monotone line search-users guide, version 2.2," Rep., Dept. of Comput. Sci., University of Bayreuth, Bayreuth, Germany, Tech. Rep., 2006.

[31] ——, "A robust implementation of a sequential quadratic programming algorithm with successive error restoration," *Optimization Lett.*, vol. 5, no. 2, pp. 283–296, May 2011.

[32] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *Proc. IEEE Conf. Intelligent Robots and Syst.*, 2012, pp. 4906–4913.

[33] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *Proc. Amer. Control Conf.*, 2005, pp. 300–306.

[34] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems." in *Int. Conf. on Informatics in Control, Automation and Robotics*, 2004, pp. 222–229.

[35] T. Albahkali, R. Mukherjee, and T. Das, "Swing-up control of the Pendubot: an impulse–momentum approach," *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 975–982, Aug. 2009.

[36] Y. Orlov, L. T. Aguilar, L. Acho, and A. Ortiz, "Swing up and balancing control of Pendubot via model orbit stabilization: Algorithm

synthesis and experimental verification," in *Proc. IEEE Conf. Decision and Control*, 2006, pp. 6138–6143.
[37] M. W. Spong and D. J. Block, "The Pendubot: A mechatronic system for control research and education," in *Proc. IEEE Conf. Decision and Control*, vol. 1, 1995, pp. 555–556.
[38] M. W. Spong, "The swing up control problem for the Acrobot," *IEEE Control Syst. Mag.*, vol. 15, no. 1, pp. 49–55, Feb. 1995.
[39] X. Xin and M. Kaneda, "Analysis of the energy-based swing-up control of the Acrobot," *Int. J. of Robust and Nonlinear Control*, vol. 17, no. 16, pp. 1503–1524, Nov. 2007.
[40] X. Xin and T. Yamasaki, "Energy-based swing-up control for a remotely driven Acrobot: Theoretical and experimental results," *IEEE Trans. Control Syst. Technol.*, vol. 20, no. 4, pp. 1048–1056, Jul. 2012.
[41] I. Fantoni, R. Lozano, and M. W. Spong, "Energy based control of the Pendubot," *IEEE Trans. Autom. Control*, vol. 45, no. 4, pp. 725–729, Apr. 2000.
[42] X.-Z. Lai, J.-H. She, S. X. Yang, and M. Wu, "Comprehensive unified control strategy for underactuated two-link manipulators," *IEEE Trans. Syst., Man, Cybern. B*, vol. 39, no. 2, pp. 389–398, Apr. 2009.
[43] M. W. Spong, "Underactuated mechanical systems," in *Control Problems in Robotics and Automation*. Springer, 1998, pp. 135–150.
[44] M. W. Spong, P. Corke, and R. Lozano, "Nonlinear control of the reaction wheel pendulum," *Automatica*, vol. 37, no. 11, pp. 1845–1851, Nov. 2001.
[45] P. Kulchenko and E. Todorov, "First-exit model predictive control of fast discontinuous dynamics: Application to ball bouncing," in *Proc. IEEE Conf. Robotics and Automation*, 2011, pp. 2144–2151.
[46] H. J. Sussmann, "A maximum principle for hybrid optimal control problems," in *Proc. IEEE Conf. Decision and Control*, vol. 1, 1999, pp. 425–430.
[47] O. Arslan, "Model based methods for the control and planning of running robots," Ph.D. dissertation, Bilkent University, Bilkent/Ankara, Turkey, 2009.
[48] B. Dadashzadeh, H. Vejdani, and J. Hurst, "From template to anchor: A novel control strategy for spring-mass running of bipedal robots," in *Proc. IEEE Conf. Intelligent Robots and Syst.*, Sept. 2014, pp. 2566–2571.
[49] J. W. Hurst, J. E. Chestnutt, and A. A. Rizzi, "The actuator with mechanically adjustable series compliance," *IEEE Trans. Robot.*, vol. 26, no. 4, pp. 597–606, Aug. 2010.
[50] M. H. Raibert, "Legged robots," *Commun. of the ACM*, vol. 29, no. 6, pp. 499–514, Jun. 1986.
[51] R. D. Gregg, A. K. Tilton, S. Candido, T. Bretl, and M. W. Spong, "Control and planning of 3-D dynamic walking with asymptotically stable gait primitives," *IEEE Trans. Robot.*, vol. 28, no. 6, pp. 1415–1423, Dec. 2012.
[52] P. A. Parrilo, "Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization," Ph.D. dissertation, Caltech, Pasadena, CA, 2005.
[53] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "LQR-trees: Feedback motion planning via sums-of-squares verification," *The Int. J. of Robotics Research*, vol. 29, no. 8, pp. 1038–1052, Apr. 2010.
[54] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Predictive active steering control for autonomous vehicle systems," *IEEE Trans. Control Syst. Technol.*, vol. 15, no. 3, pp. 566–580, May 2007.
[55] L. Mejias, S. Saripalli, P. Campoy, and G. S. Sukhatme, "Visual servoing of an autonomous helicopter in urban areas using feature tracking," *J. of Field Robotics*, vol. 23, no. 3-4, pp. 185–199, Mar. 2006.
[56] R. Schroer, "Flight control goes digital [part two, NASA at 50]," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 23, no. 10, pp. 23–28, Oct. 2008.
[57] M. Turpin, N. Michael, and V. Kumar, "Trajectory design and control for aggressive formation flight with quadrotors," *Autonomous Robots*, vol. 33, no. 1-2, pp. 143–156, Aug. 2012.
[58] R. Gregg, T. Bretl, and M. Spong, "A control theoretic approach to robot-assisted locomotor therapy," in *Proc. IEEE Conf. Decision and Control*, Dec 2010, pp. 1679–1686.
[59] S. Jezernik, G. Colombo, and M. Morari, "Automatic gait-pattern adaptation algorithms for rehabilitation with a 4-DOF robotic orthosis," *IEEE Trans. Robot. Autom.*, vol. 20, no. 3, pp. 574–582, Jun. 2004.
[60] H. Kazerooni, A. Chu, and R. Steger, "That which does not stabilize, will only make us stronger," *The Int. J. of Robotics Research*, vol. 26, no. 1, pp. 75–89, Jan. 2007.
[61] A. Mavrommati, A. R. Ansari, and T. D. Murphey, "Optimal control-on-request: An application in real-time assistive balance control," in *Proc. IEEE Conf. Robotics and Automation*, 2015, pp. 5928–5934.
[62] K. N. Winfree, P. Stegall, and S. K. Agrawal, "Design of a minimally constraining, passively supported gait training exoskeleton: ALEX II," in *IEEE Int. Conf. Rehabilitation Robotics*, 2011, pp. 1–6.
[63] A. J. Barry, "Flying between obstacles with an autonomous knife-edge maneuver," Master's thesis, MIT, Cambridge, MA, 2012.
[64] A. J. Barry, T. Jenks, A. Majumdar, H.-T. Lin, I. G. Ros, A. Biewener, and R. Tedrake, "Flying between obstacles with an autonomous knife-edge maneuver," In *Proc. IEEE Conf. Robotics and Automation*, Video Track, 2014.
[65] C. Laschi, M. Cianchetti, B. Mazzolai, L. Margheri, M. Follador, and P. Dario, "Soft robot arm inspired by the octopus," *Advanced Robotics*, vol. 26, no. 7, pp. 709–727, Jan. 2012.
[66] R. F. Shepherd, F. Ilievski, W. Choi, S. A. Morin, A. A. Stokes, A. D. Mazzeo, X. Chen, M. Wang, and G. M. Whitesides, "Multigait soft robot," *Proc. Nat. Academy of Sciences*, vol. 108, no. 51, pp. 20 400–20 403, Dec. 2011.
[67] L. Pronzato, "Optimal experimental design and some related control problems," *Automatica*, vol. 44, no. 2, pp. 303–325, Feb. 2008.
[68] A. D. Wilson, J. A. Schultz, and T. D. Murphey, "Trajectory optimization for well-conditioned parameter estimation," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 1, pp. 28–36, Jan. 2015.
[69] A. D. Wilson, J. A. Schultz, A. R. Ansari, and T. D. Murphey, "Real-time trajectory synthesis for information maximization using sequential action control and least-squares estimation," in *Proc. IEEE Conf. Intelligent Robots and Syst.*, 2015, pp. 4935–4940.
[70] B. D. O. Anderson and J. B. Moore, *Optimal control: linear quadratic methods*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1990.
[71] H. K. Khalil, *Nonlinear systems*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996.
[72] J. P. Hespanha, *Linear systems theory*. Princeton, New Jersey, USA: Princeton University Press, 2009.
[73] M. Diehl, R. Amrit, and J. B. Rawlings, "A Lyapunov function for economic optimizing model predictive control," *IEEE Trans. Autom. Control*, vol. 56, no. 3, pp. 703–707, Mar. 2011.
[74] L. Grüne and J. Pannek, *Nonlinear model predictive control*. London, UK: Springer, 2011.
[75] A. Jadbabaie and J. Hauser, "On the stability of receding horizon control with a general terminal cost," *IEEE Trans. Autom. Control*, vol. 50, no. 5, pp. 674–678, May 2005.
[76] J. H. Lee, "Model predictive control: review of the three decades of development," *Int. J. of Control, Automation and Syst.*, vol. 9, no. 3, pp. 415–424, Jun. 2011.
[77] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, Jun. 2000.
[78] H. Gonzalez, R. Vasudevan, M. Kamgarpour, S. S. Sastry, R. Bajcsy, and C. J. Tomlin, "A descent algorithm for the optimal control of constrained nonlinear switched dynamical systems," in *ACM Conf. on Hybrid Syst.: Computation and Control*, 2010, pp. 51–60.