Model-Based Reactive Control for Hybrid and High-Dimensional Robotic Systems

Emmanouil Tzorakoleftherakis, Alex Ansari, Andrew Wilson,
Jarvis Schultz, and Todd D. Murphey

***Abstract*—Sequential action control (SAC) is a recently developed algorithm for optimal control of nonlinear systems. Previous work by the authors demonstrates that SAC performs well on several benchmark control problems. This work demonstrates applicability of SAC to a variety of robotic systems; we show that SAC can also be easily applied to hybrid systems without any modification and that its scalability facilitates application to high-dimensional systems. First, SAC is applied to a popular hybrid dynamic running model known as the spring-loaded inverted pendulum (SLIP). The results show that SAC can achieve dynamic hopping without using prescribed touchdown angles/leg stiffness. Moreover no specialized hybrid methods are necessary to handle the contact dynamics, despite the nonsmooth nature of the problem. The same SAC-controlled SLIP model is also implemented in a game for the Android operating system, demonstrating the minimal computational requirements for implementing SAC. Our second example involves successful stabilization and tracking control of a nonlinear, constrained dynamic model of a humanoid marionette with 56 states and 8 inputs. Finally, a discussion that includes best practices on tuning parameters of the SAC algorithm as well as the challenges of hardware implementation is also provided, along with a video that shows the resulting simulations for each example.**

***Index Terms*—Optimization and optimal control, underactuated robots, simulation and animation**

I. INTRODUCTION

SEQUENTIAL action control (SAC) is a model-based control approach that provides closed-loop optimal actions for nonlinear systems from a closed-form expression. Derivations and examples in [1]–[3] show that the method takes advantage of dynamics and develops constrained optimal actions on-line that outperform off-line nonlinear trajectory optimization on benchmark control problems. While these examples represent challenging and well-understood control tasks, they are limited to smooth systems of state dimension ≤ 8.

This paper shows that SAC may be successfully applied to both *hybrid* and *high-dimensional* systems. Current Nonlinear Model Predictive Control (NMPC) approaches that have been applied to these types of systems include differential dynamic programming (DDP) and linear quadratic regulator/gaussian synthesis (LQR/LQG) [4]–[7]. These methods are based on linear approximations of the dynamic system (second-order in the case of DDP) and usually assume quadratic objective functions or second-order approximations of the cost. They derive optimal controls indirectly by translating a numerically challenging, $2\times n$-dimensional (n is the dimension of the state vector) two-point boundary value problem (TPBVP), to a symmetric $n \times n$ system of Riccati equations. In contrast to these alternatives, controls in SAC have an analytic form, which significantly reduces execution time. Instead of iteratively minimizing quadratized objectives, SAC seeks to *improve* the *nonlinear* objective at each iteration. The approach is well-posed and does not need to address positive-definiteness of the quadratized cost, unlike e.g. [8]. A more detailed comparison of SAC with the NMPC literature is given in Section II-B.

Modeling contact dynamics and incorporating constraints on the control values are two additional challenges in control of dynamic systems. While many (online) trajectory optimization routines often require specialized and computationally expensive methods to handle these cases [9]–[13], control actions in SAC can be directly saturated without additional calculations [2]. Moreover, we show here that SAC can handle contact dynamics without any modification to the algorithm or additional computational cost.

The contribution of this paper is three-fold. First, we show that SAC's algorithmic approach can automate control for a variety of robotic systems, including those that locomote using impacts. In particular, we apply SAC in simulation to a popular dynamic running model known as the spring-loaded inverted pendulum (SLIP). The SLIP is a nonlinear underactuated hybrid system with impacts that provides a relatively low-dimensional representation of the center of mass dynamics and energetics of running for a wide variety of animal species [14]–[16]. A number of robots have been designed according to this model [17]–[19], while others use it as a template for control [17], [20]–[23].

Controllers for the SLIP hopper and related dynamic locomotion models often utilize feed-forward gaits/body trajectories, prescribed touchdown angles/leg stiffness (often controlled by dead-beat approaches) or (numerical) approximations, e.g. of the solution to the stance dynamics or the return map, designed to account for some degree of terrain variation [17], [20]–[22], [24], [25]. While these methods are advantageous in that they do not require a terrain model, for robots designed to directly emulate the SLIP [17], [19],

Manuscript received: August, 31, 2015; Revised December, 6, 2015; Accepted January, 16, 2016.

This paper was recommended for publication by Editor Dr. Kevin Lynch upon evaluation of the Associate Editor and Reviewers' comments. The material in this paper is based upon work supported by the National Science Foundation under Grants CNS 1329891 and CMMI 1334609.

Authors are with the Neuroscience and Robotics Laboratory (N×R), Department of Mechanical Engineering, Northwestern University, 2145 Sheridan Road, Evanston, IL 60208, USA.

Email: man7therakis@u.northwestern.edu, alex2011@u.northwestern.edu, awilson@u.northwestern.edu, jschultz@northwestern.edu, t-murphey@northwestern.edu

Digital Object Identifier (DOI): see top of this page.

these processes ignore potential collisions (e.g. ledges between stairs) by not accounting for the motion of the swing-leg between take-off and landing. Online feedback control over varied terrain is generally challenging, and some successful methods rely on several decoupled feedback control mechanisms working in tandem [16], [23], [26]. This approach, although effective, is more complex since several control loops have to be designed and tuned to work in parallel. In contrast, we demonstrate that SAC can compute constrained closed-loop controls in real time to steer the SLIP hopper across varied terrain, without requiring a pre-designed trajectory gait, explicit control of the touchdown angles/leg stiffness or any approximations. Moreover, this is achieved based on high-level trajectory goals specifying the desired direction of motion for the SLIP center of mass, without using decoupled control loops. Despite the nonsmooth nature of the problem, the SAC algorithm does not deviate from its normal execution to incorporate specialized solvers unlike many traditional trajectory optimization/optimal control methods [9]–[12], thus avoiding additional computational overhead. Moreover, we show that SAC can handle rugged terrain like stairs while accounting for swing-leg motion.

The second contribution is an Android application that implements the SAC-controlled SLIP model in a 2-D game. This example suggests that real-time, nonlinear optimal control is possible, even on devices with limited computational resources. Lastly, we demonstrate that our approach can compute trajectories for closed-loop pose control of high dimensional systems. Specifically, we use an underactuated 56-state marionette in our simulation, that was modeled using `trep` [27]—an open-source software simulation package. A discussion that includes best practices on tuning parameters of the SAC algorithm as well as the challenges of hardware implementation is also provided to bridge the gap between theory and application and to encourage the reader to utilize our methods.

II. Sequential Action Control

In this paper we demonstrate the applicability of SAC, a recently formulated algorithm for control of general nonlinear systems, to hybrid and high-dimensional systems. For convenience, we will now briefly summarize the algorithm presented in [1]–[3].

SAC enables rapid, closed-loop constrained control synthesis for broad range of systems and objectives. The systems controlled by SAC are assumed to be in linear-affine form, i.e. nonlinear with respect to the state vector, $x \in \mathbb{R}^n$ and linear (or linearized) with respect to the control vector, $u \in \mathbb{R}^m$, such that

$$\dot{x} = f(x,u) = g(x) + h(x)\,u. \tag{1}$$

As opposed to many methods, SAC is not restricted to a linear quadratic cost. It applies to general tracking objectives of the form

$$J_{track} = \int_{t_0}^{t_0+T} l(x(t))\,dt + m(x(t_0+T))\,, \tag{2}$$

with differentiable incremental cost $l(x(t))$ and terminal cost $m(x(t_0+T))$. Although (2) lacks a norm on control effort, SAC includes this norm in the following step, in (4). It should be noted that SAC is not specific to trajectory tracking; energy tracking as in [28] or tracking a "point" as shown in the SLIP simulation in Section III-A is also possible. As a result of its control synthesis process, SAC can calculate controls that optimally improve (2) even in the situations where the objective is non-convex or unbounded. The SAC algorithm follows a cyclic, closed-loop process illustrated in Fig. 1. As the cycle iterates, SAC sequences together a piecewise continuous closed-loop response (see the SAC action signal at the bottom of Fig. 1). Beginning with prediction, the major steps of the algorithm are described in the following subsections.

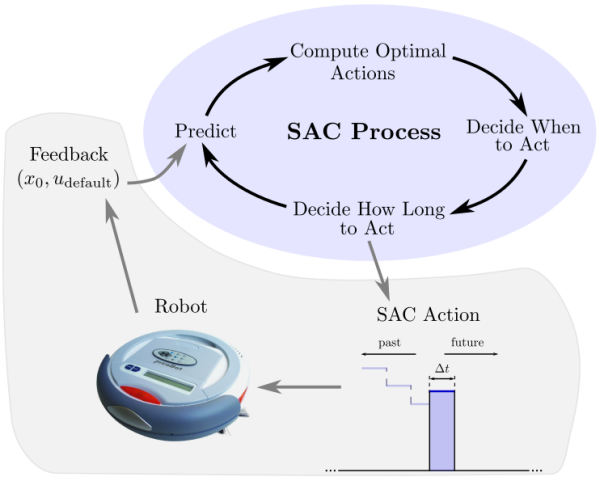

Fig. 1. An overview of the SAC control process.

A. SAC Steps

1) Predict: The SAC process begins by predicting the evolution of a system model from current state feedback. In this step, the algorithm simulates the system (1) from the current state x_0 and time t_0, for the finite horizon $[t_0,\ t_0+T]$, under a default (nominal) control $u = u_{\text{default}}$. The horizon length T, is a design parameter. Without loss of generality, the default control throughout this paper is null, $u_{\text{default}} \triangleq 0$. The term is included in formulas for completeness and indicates potential shared control implementation. As an example, u_{default} may be an optimized feedforward controller providing a nominal trajectory around which SAC would provide feedback.

The sensitivity of (2) to state variations along the predicted trajectory is provided by an adjoint variable, $\rho \in \mathbb{R}^n$, also simulated during the prediction step. The adjoint satisfies

$$\begin{aligned} \dot{\rho} &= -D_x l(x)^T - D_x f(x, u_{\text{default}})^T \rho \\ \text{subject to } \rho(t_0+T) &= D_x m(x(t_0\ +\ T))^T. \end{aligned} \tag{3}$$

The prediction phase completes upon simulation of the state and the adjoint system under u_{default} control. The resulting trajectories $x(\cdot)$, $\rho(\cdot)$ will be used in (4) in the following section.

2) Compute optimal actions: Each iteration of the SAC process loop depicted in Fig. 1 returns a set of values for the control vector, the control application time (Section II-A3) and

the control duration (Section II-A4). A single vector of control values along with its associated application time and duration define a SAC control *action* as produced at each iteration.

Before computing the control application time and duration, SAC computes a schedule, $u^* : \{t \,|\, t \in [t_0, t_0 + T]\} \mapsto \mathbb{R}^m$, corresponding to the values of the control action that would optimally improve performance if applied for some duration at an arbitrary time $t \in [t_0, t_0 + T]$. The control action values in u^* optimize

$$J_u = \frac{1}{2} \int_{t_0}^{t_0+T} \left[\frac{dJ_{track}}{d\lambda} - \alpha_d \right]^2 + \|u(t)\|_R^2 \, dt \,,$$
$$\text{with } \frac{dJ_{track}}{d\lambda} = \rho(t)^T (f(x(t), u) - f(x(t), u_{\text{default}})). \quad (4)$$

The quantity $\frac{dJ_{track}}{d\lambda}$ (see [29]) denotes the rate of change of the cost with respect to a switch of infinitesimal duration λ in dynamics, produced by a SAC action applied at some time τ. Thus, $\frac{dJ_{track}}{d\lambda}$ intrinsically parameterizes an action by its application time and duration, which is why these two variables are calculated in the cyclic process in Sections II-A3 and II-A4. Intuitively, SAC is improving the open-loop cost (2) by driving $\frac{dJ_{track}}{d\lambda}$ to a negative value $\alpha_d \in \mathbb{R}^-$ through minimization of (4). This parameter, α_d, is user specified and allows the designer to influence how aggressively each control action improves the current trajectory cost.

Based on the simulation of the dynamics (1), and (3) completed in the prediction step (Section II-A1), the control schedule that minimizes (4) is provided as a closed-form expression,

$$u^* = (\Lambda + R^T)^{-1} \left[\Lambda \, u_{\text{default}} + h(x)^T \rho \, \alpha_d \right] , \quad (5)$$

with $\Lambda \triangleq h(x)^T \rho \rho^T h(x)$.

3) Decide when to act (find τ): As mentioned before, the quantity $\frac{dJ_{track}}{d\lambda}$ parameterizes an action by its application time τ. As a result, the SAC algorithm optimizes a decision variable not normally included in control calculations—the choice of *when* to act. The curve u^* provides the values of possible actions that SAC could take at different times to optimally improve system performance from that time. The algorithm chooses one of these actions to apply at each iteration of the SAC process and then re-computes the curve u^* from current state feedback at the next iteration. In choosing when to act (choosing an action from curve u^*), SAC searches u^* for a time τ that optimizes the trade-off between the cost of waiting and the efficacy of control at that time according to,

$$J_t(\tau) = \|u^*(\tau)\| + \left. \frac{dJ_{track}}{d\lambda} \right|_\tau + (\tau - t_0)^\beta \,. \quad (6)$$

The parameter $\beta \in \mathbb{R}$ is usually chosen to be a fixed value, $\beta \in [1, 2]$, encoding the cost of waiting.

4) Decide how long to act (find λ): The quantity $\frac{dJ_{track}}{d\lambda}$ also parameterizes an action by its duration. After computing the values of potential optimal actions from (5) and choosing when to act based on (6), the final step in synthesizing a SAC action is to choose how long to act (select the control duration). It is typically assumed that actions will last for short duration as the control synthesis cycle is fast and the next action is prepared for implementation quickly. For these reasons, SAC implementations apply a *line search* process. Starting with a (short) initial duration, $\lambda = \lambda_0$, the effect of the control action is simulated from (1) and (2). If the simulated action improves cost (2), the duration is selected. If this is not the case, the duration is reduced and the process is repeated.

After computing the duration, λ, the SAC action is fully specified (it has a value, an application time and a duration). As an additional step, when $u_{\text{default}} = 0$, actions can be directly saturated to satisfy any min/max control constraints of the form $u_{min,k} < 0 < u_{max,k} \; \forall k \in \{1, \ldots, m\}$ (see [2] for a proof that saturated controls still result in a reduction in cost). By iterating on this process (Section II-A1 until Section II-A4), SAC rapidly synthesizes piecewise-continuous, constrained control laws for nonlinear systems. For more information about SAC, the reader is encouraged to consult [1]–[3].

B. Comparison to NMPC Literature

The following points are worth noting when comparing SAC to alternative NMPC methods (see e.g. [4]–[7], [30]–[33] and references therein).

1) SAC uses the continuous-time dynamics, thus allowing for variable-step integration, as opposed to many NMPC alternatives that utilize the discrete-time dynamics [7], [30], [31], [33], [34].

2) SAC is applied to the *nonlinear* cost function as opposed to e.g. DDP and LQR/LQG approaches in [4]–[7] which use quadratic approximations of the cost.

3) When solving the open-loop problem at each iteration (see steps in Sections II-A1 through II-A4), SAC does not minimize but rather *improves* the nonlinear cost function. On the other hand, many NMPC methods compute an open-loop optimal control signal over a finite horizon $[t_0, t_0 + T]$ (usually over the entire interval), to *minimize* the objective on that interval. As the horizon window changes, the calculated control is applied for the interval $[t_0, t_0 + t_s]$ (where t_s is the sampling time) and the remaining control for the interval $[t_0 + t_s, t_0 + T]$ is discarded or used to seed the control optimization on the next interval $[t_0 + t_s, t_0 + t_s + T]$. However, the computed control on the interval $[t_0, t_0 + T]$ only guarantees reduction in cost when the *entire* time horizon is used, even though the control signal is only applied on the interval $[t_0, t_0 + t_s]$. Moreover, minimizing a quadratic approximation of the cost as in DDP and LQR/LQG methods, does not guarantee that the solution is a (local) minimizer of the original nonlinear cost. As opposed to these methods, SAC computes control actions that optimally improve the cost on the entire horizon, $[t_0, t_0 + T]$, assuming that the control will be applied during the period $[t_0, t_0 + t_s]$. Thus, no portion of the calculated open-loop action is discarded.

4) The solution of the open-loop problem in SAC has an analytic form given in (5) which requires only the n-dimensional simulations of (1) and (3) ($2n$ total). Alternatively, NMPC methods either employ nonlinear programming solvers (see [35] and [36] for a review) or solve a symmetric $n \times n$ matrix of Riccati equations as, for example, in [4]–[7].

5) Control saturations can be incorporated without additional computational overhead in the SAC process (see [2] for

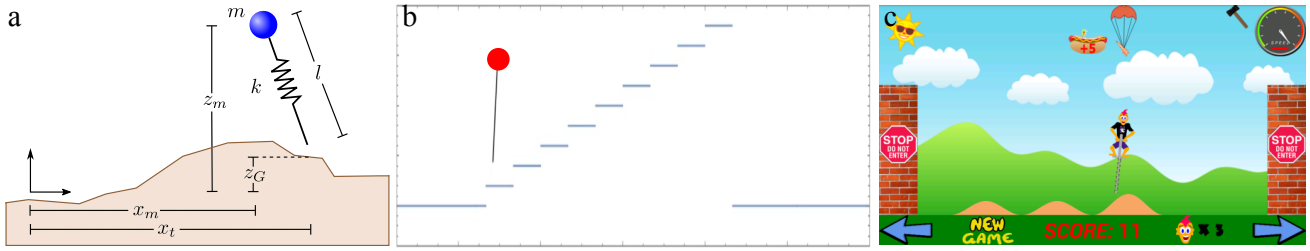

Fig. 2. a) Parameters and configuration variables for the planar SLIP system. b) Illustration of the planar SLIP successfully hopping up stairs and over sinusoidal terrain (also be seen in the accompanying video). The SAC algorithm computes constrained motions on-line to place the toe of the SLIP model and develops constrained thrusts that allow it to hop without falling. The process is automated in that it requires only a robot model and high-level trajectory goals specifying the desired direction of motion for the SLIP center of mass. c) A snapshot of an Android game (available for download) that utilizes the SLIP model.

a proof). Constraints on the state may be added by introducing penalty terms in the cost function.

6) SAC can be applied to hybrid systems without any modification. Despite the nonsmooth nature of these systems, the SAC algorithm does not deviate from its normal execution to incorporate specialized solvers unlike many traditional trajectory optimization/optimal control methods, thus avoiding additional computational overhead (see [9]–[12]).

7) SAC optimizes application time τ and control duration λ. These two decision variables, which are not normally included in alternative NMPC methods, allow more flexibility in the control calculation. For example, as explained in [2], τ can be used to avoid acting on singular configurations where control could be less effective or not effective at all (e.g. the horizontal configuration in a cart-pendulum system).

8) SAC may be computed quickly and efficiently. This follows immediately from the previous points.

Stability Remarks: It is shown in [2] that, under certain assumptions, SAC solutions simplify to linear time-varying state feedback laws near equilibrium points. Additionally, if (2) is quadratic and nominal control $u_{default}$ is modeled as applying consecutively computed optimal actions, SAC actions (5) simplify to finite horizon LQR controls [37]. In this case one can prove the existence of a Lyapunov function and guarantee local stability for SAC using methods from linear systems theory. In its current state of development, SAC lacks global guarantees for stability. However, we believe that, similar to NMPC methods, stability for SAC can be achieved by applying a terminal cost/region approach as in [30]–[33] and we have left these developments for future work.

III. Example Simulations

In this section we present two simulation examples that demonstrate the applicability of SAC to a) hybrid and b) high-dimensional systems. In the hybrid system example SAC is used to automate dynamic locomotion of a hopping mechanism over uneven terrain. We show that, unlike many trajectory optimization routines, SAC can control this model without interrupting its normal execution to include hybrid or other specialized methods. The high-dimensional example involves control of a highly nonlinear and constrained humanoid marionette model with 56 states and 8 inputs. These examples indicate potential application of SAC to a wide variety of systems.

A. Hybrid System Example: The SLIP Model

In this section we apply SAC to an underactuated nonlinear hybrid system with impacts. In particular, we utilize a spring-loaded inverted pendulum (SLIP) model; a model that is common in analysis and control synthesis of dynamic hopping and running. Figure 2a shows the configuration of the model, which consists of a point mass attached to a spring. The choice of configuration variables is similar to that in [38]. The state, $x = [x_m, \dot{x}_m, z_m, \dot{z}_m, x_t]$, consists of the 2-D position and velocities of the center of mass followed by the "toe" x coordinate, x_t, corresponding to the spring endpoint.

The dynamics of the SLIP are hybrid and include two modes / phases: 1) a flight phase where the toe endpoint is in the air and 2) a stance phase where the toe is in rolling contact with the ground. The length, L, of the SLIP model matches the resting length of the spring, $L = L_0$, in flight and

$$L = \sqrt{(x_m - x_t)^2 + (z_m - z_G)^2} \tag{7}$$

in the stance phase. The z_G term in (7) tracks the height of the terrain at the location of the toe. Note that (7) assumes the model is in the stance phase where z_G is the height of the toe. The transition between flight and stance is state-based and determined by zero crossings of an indicator function,

$$\phi(x) = z_m - \frac{L_0(z_m - z_G)}{L} - z_G\,, \tag{8}$$

which applies L from (7). Hence, we assume the spring is in compression when in stance and the model lifts-off once it expands back to full (rest) length.

Control authority for the SLIP also switches with phase. On the ground SAC can apply force along the spring axis, u_s, and in flight SAC can directly control the velocity of the toe u_f along the x axis. The complete control vector is $u = [u_s, u_f]^T$. The flight phase dynamics,

$$f_f(x,u) = \begin{pmatrix} \dot{x}_m \\ 0 \\ \dot{z}_m \\ -g \\ \dot{x}_m + u_f \end{pmatrix}, \tag{9}$$

and stance phase dynamics,

$$f_s(x,u) = \begin{pmatrix} \dot{x}_m \\ \frac{(k(L_0-L)+u_s)(x_m-x_t)}{ml} \\ \dot{z}_m \\ \frac{(k(L_0-L)+u_s)(z_m-z_G)}{ml} - g \\ 0 \end{pmatrix}, \qquad (10)$$

depend on gravity, $g = 9.81\text{m/s}^2$, mass, $m = 1\,\text{kg}$, a spring resting length, $L_0 = 1\,\text{m}$, and spring constant, $k = 100\frac{\text{N}}{\text{m}}$. The simulation was based on the following parameter values: we used a quadratic trajectory cost based on (2) with $l(x(t)) = \frac{1}{2}(x(t) - x_d)^T Q\,(x(t) - x_d)$ and $m(x(t_f)) = 0$, with $Q = Diag[\,0\,,70\,,50\,,0\,,0\,]$, and time horizon $T = 0.6$s. Parameters of the control cost in (4) were selected as $R = 0.1$, $\alpha_d = \gamma J_{track}$ with proportional feedback constant $\gamma = -10$. Finally, control inputs were constrained on-line so that $|u_f| \leq 5\,\frac{\text{m}}{\text{s}}$ and $|u_s| \leq 30\,\text{N}$.

For a 90s closed-loop trajectory on varying terrain (30 Hz feedback), the SLIP requires $<$ 2s to simulate on a laptop. As the trajectory results in Fig. 3 (corresponding to the stair climbing example in Fig. 2b) show, SAC uses the flight dynamics (9) to include and automate swing-leg planning—no prescribed leg stiffness/touchdown angles or other approximations are used. SAC provides closed-loop, velocity constrained swing-leg motions on-line that avoid ledges and uses thrust to hop up stairs (or uneven terrain as in Fig. 2c). These tasks are achieved based on high-level trajectory goals specifying the desired direction and height of motion for the SLIP center of mass. The latter can be typically determined by sensors such as vision, radar, and laser, perhaps combined with pre-defined maps, generating a model of the terrain ahead. In the stair-climbing simulation we used $x_d = [\,0\,,0.7\,\frac{\text{m}}{\text{s}}\,, z_G + 1.4\,\text{m}\,,0\,,0\,]^T$ as our trajectory goal and the terrain model, z_G, was prescribed by a piecewise function with a rise of 0.2 m every $\frac{2}{3}$ m. This choice of x_d specifies the same constant velocity translation at fixed desired height.

Even with the hybrid dynamical model, the SAC algorithm does not deviate from its normal execution to handle contact dynamics, thus avoiding additional computational overhead. This is in contrast with many existing trajectory optimization/optimal control methods (e.g. [9]–[12]), that do not normally accommodate nonsmooth events and require the use of specialized algorithms to do so.

Note that we have also implemented the planar version of the SLIP as an Android game (see a snapshot of the app in Fig. 2c) that is available for free download at nxr.northwestern.edu/sites/default/files/files/SACGames.apk. In the Android game, the user provides a reference velocity for the hopper along the horizontal axis using the phone's accelerometer. The SAC algorithm uses the same parameters as in the stairs example and the sinusoidal floor corresponds to ground height $z_G(x) = 0.2\cos(4x) + 0.2$ m for the first two "bumps" and $z_G(x) = 0.3\cos(4x) + 0.3$ m for the last.

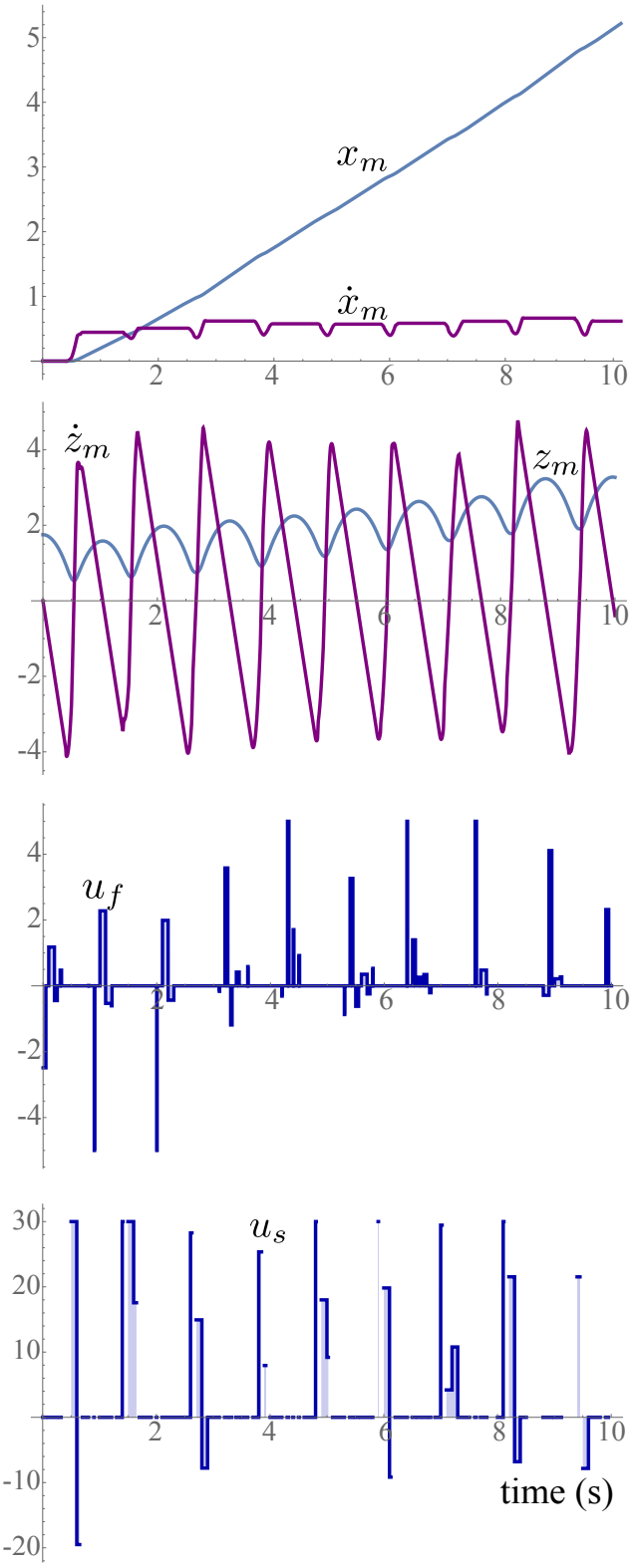

Fig. 3. Trajectory corresponding to the planar SLIP hopping up stairs (Fig. 2b).

B. High Dimensional Example: Humanoid Marionette Pose Control

To illustrate the scalability of SAC control synthesis, we utilized SAC to control a simulation of an underactuated, constrained, and highly nonlinear humanoid marionette model with 56 states, 4 string length constraints. and 8 control inputs. The controls are two in-plane forces for each of the four string endpoints (see green points in Fig. 4a, right). Two

different control tasks are presented – stabilization of the marionette to an equilibrium point and tracking of predefined trajectories for the arms of the marionette. Note that our model utilizes generalized coordinates for the system and the `trep` software package is used for calculating the forward dynamics as well as first order linearizations [27]. `Trep` (see `nxr.northwestern.edu/trep`) allows for dynamic simulation of arbitrary mechanical systems in generalized coordinates described using a tree structure. The generalized coordinates used in this model result in a conservative number of states; other choices can lead to as many as 124 states (9 rigid bodies $\in SO(3)$ and four string endpoints with two degrees-of-freedom each).

The simulation was based on the following parameter values: we used a quadratic trajectory cost based on (2) with $l(x(t)) = \frac{1}{2}(x(t) - x_d)^T Q\,(x(t) - x_d)$ and $m(x(t_f)) = 0$, with the entries of Q and R based on standard values (scaling of the identity matrix) for both stabilization and tracking. The time horizon was set to $T = 1.0$s and similar to the SLIP simulation, $R = 0.1$, $\alpha_d = \gamma J_{track}$ with proportional feedback constant $\gamma = -10$. Finally control saturation constraints were arbitrarily specified as ± 1 N, along with the SAC loop frequency of 20 Hz.

Stabilization: Stabilization involved moving the marionette model from an initial pose with the left arm raised, to a desired equilibrium pose x_d, specified as the origin (see top right panel in Fig. 4a). We compared trajectories generated by SAC with the uncontrolled system trajectories (free dynamics). The simulation results are shown in Fig. 4b. It is clear that SAC stabilized the system to the origin quickly compared to the free motion of the marionette. The SAC-controlled simulation for the stabilization took $155s$ to compute the $3s$ trajectory—about $50\times$ slower than real-time.

Tracking: The tracking task used SAC to track predefined trajectories for the arms and shoulders. In particular, we used a time-parameterized sine wave as the reference configuration for one of the degrees of freedom for both the left and right shoulder joints. This reference has the marionette cyclically spreading and closing its arms. All other reference configurations were set to their equilibrium values, and the reference velocities were all zero. Note that this reference is dynamically infeasible – perfect tracking cannot be expected. The simulation took $200s$ for a $10s$ trajectory (about $20\times$ slower than real-time). The resulting configuration trajectories are shown in Fig. 4c. Even for this large-dimensional, highly underactuated system, SAC managed reasonable tracking performance of the two moving shoulder joints with little excitation of the other configurations. For a better illustration of this example see the accompanying video.

While the marionette example is not yet running in real time, we note that the current implementation relies on interpreted run-time code that would be straightforward to precompile. Combined with parameter optimization, we expect real-time control for systems of similar size is possible on standard computing hardware.

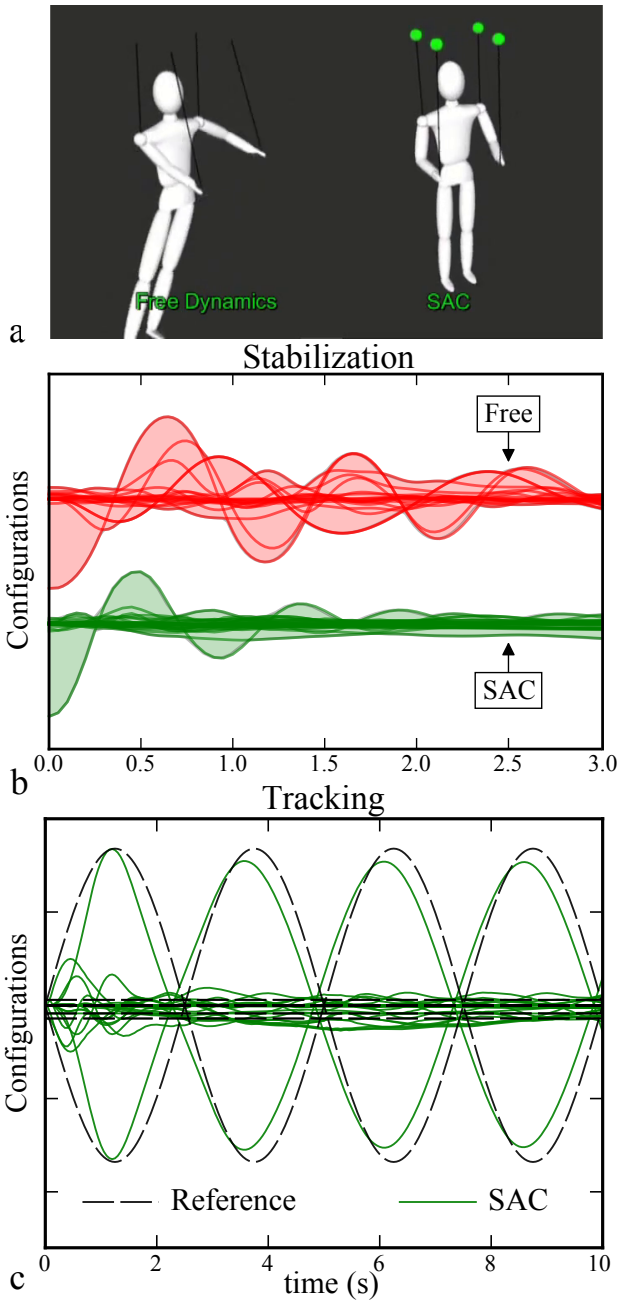

Fig. 4. a) SAC controls the pose of a constrained and underactuated nonlinear marionette model with 56 states, 4 string length constraints, and 8 control inputs based on a real robot-controlled marionette system [39]. Modeling is done using the `trep` software [27]. b) The SAC controller uses a quadratic objective to compute constrained controls in closed-loop that drive four strings endpoints (green), such that the marionette transitions to a desired configuration (stabilization at the origin). Note that the free dynamics (red) are very lightly damped as seen in the accompanying video. c) Configuration trajectories for the marionette tracking control task generated by SAC. The shoulder joints track the predefined sine wave while the rest of the body remains stationary. The tracking motion can also be viewed in the accompanying video.

IV. DISCUSSION

A. Tuning SAC Parameters

Besides a model of system dynamics the only required modifications for applying SAC to varying systems are the

encoding of the objective in (2), and tuning the parameters of the algorithm. SAC parameters that must be tuned include the time horizon T and the "controller aggressiveness" α_d (see (4)). Other parameters that may appear and require tuning include the control saturation limits, cost weighting matrices (e.g. Q and R) and loop/feedback frequency.

Trajectory cost: Both presented cases (SLIP and marionette) use a quadratic trajectory cost requiring a Q matrix and desired trajectory, x_d. In practice, the SAC algorithm does not appear to be very sensitive to changes in the weighting matrices – generally much less sensitive than traditional linear-quadratic regulator design. Consequently, once good values for these parameters are found, it is rare to have to adjust the values. As evidence, for the marionette system, the entries of Q and R are based on standard values (scaling of the identity matrix) for stabilization and tracking, while for the SLIP, both examples apply the same Q and R.

Control saturation: These values are generally selected off-line and do not require further tuning.

Time horizon T and aggressiveness α_d: Parameters T and α_d may require tuning to accommodate different system models. Similar to the weight matrices in (2) however, once specified, they often result in reasonable performance for a variety of conditions and tracking objectives. As evidence, both SLIP examples use the same values with $T = 0.6$s and $\alpha_d = \gamma J_{track}$ with proportional feedback constant $\gamma = -10$. As mentioned before, for a 90s closed-loop trajectory on varying terrain (30 Hz feedback), the SLIP requires $<$ 2s to simulate on a laptop. Thus, similar to existing methods designed around empirically stable SLIP hopping (see the background section in [40]), we leveraged the speed of SAC to search for parameters that yield successful long-term hopping over varying terrain and initial conditions. As in the SLIP case, both presented problems in the marionette simulation (stabilization and tracking) used the same values for T and a_d. Intuitively, systems with slower dynamics (e.g. a long pendulum) will typically need higher T values to follow a trajectory based on state error.

Loop frequency: The importance of this parameter is explained in the following section. In general, high loop frequency is preferable since it allows the algorithm to incorporate feedback faster, which is crucial when dealing with uncertainty.

B. Challenges of Hardware Implementation

Hardware implementation is often challenging, regardless of the control algorithm. For SAC, there is a limited number of experimental applications to date [41] since the algorithm was only recently formulated (this is the focus of our current work). Nevertheless, there are a few points that should be highlighted when implementing SAC on a real system.

Looking at Fig. 3 and previous examples in [1]–[3], it is clear that SAC generates discontinuous control signals. However, many actuators cannot generate signals of this form—the result of forcing the actuator to track the discontinuous SAC output would be a smoothed version of that control signal, which would most likely fail to meet the performance requirements of the system. It must be noted that situations like this, where the actuators cannot reliably generate the calculated control input, are not SAC-specific and can be found in other algorithms as well, perhaps for different reasons (e.g. control saturation). In our case, a solution to this problem would be to calculate SAC actions in an appropriate control space. For example, to control the cart velocity in the cart pendulum system, one could set SAC to control the cart acceleration (note that this approach increases the dimensionality of the state space by m in the worst case scenario). When the SAC-calculated acceleration is integrated, the corresponding velocity would be continuous and thus feasible for the actuators.

Uncertainty is also an issue in an experimental setup. Common sources of uncertainty include noise in measurements and model discrepancies. Nevertheless, one of the benefits of MPC is the use of feedback to partially compensate for these types of uncertainty. Naturally, better results can be achieved by using high frequency feedback. SAC in particular is well-suited for this scenario, since as described in Section II-B, control calculations are generally fast. In the event that high frequency feedback/communication is not supported by the hardware, then noise and model inaccuracies could be critical for all (MPC) algorithms. In that case, time-critical, SAC-specific processes like the calculation of the application time and duration of an action can be adjusted so as to provide solutions in agreement with the attainable bandwidth.

V. Conclusions and Future Work

This paper utilizes the computational advantages of sequential action control (SAC) for control policy generation. Simulations demonstrate the approach on a high-dimensional system (a 56-state marionette model) and in hybrid dynamical locomotion (using a spring-loaded inverted pendulum – SLIP). In the particular case of the SLIP, the fact that SAC controls the system with little domain/system specific knowledge is noteworthy. The algorithm is also applied with little modification to accommodate the hybrid nature of the SLIP model. These examples, as well as the benchmark examples presented in [1]–[3], demonstrate potential applications of SAC to a wide variety of systems.

Several promising research directions have been identified to help improve the applicability and utility of SAC. As mentioned in Sections II-B and IV-B developing global guarantees for stability and validating SAC experimentally are some of our immediate goals. Another likely direction for future work related to the previous ones is automated SAC parameter selection. Currently, one can find SAC parameters to provide local stability around equilibrium based on the analytical expression for optimal actions [1], [2]. However, away from equilibrium tuning may be necessary to develop long-term, empirically stable trajectories. Leveraging the computational efficiency of SAC synthesis, numerical methods (e.g. Sum-of-Squares [42]) can select parameters that provide conservative approximations of stable regions of attraction for general (nonlinear) systems.

REFERENCES

[1] A. Ansari and T. Murphey, "Control-on-request: Short-burst assistive control for long time horizon improvement," in *American Control Conference (ACC)*, 2015, pp. 1173–1180.

[2] A. Ansari and T. D. Murphey, "Sequential Action Control: Closed-form optimal control for nonlinear systems," *IEEE Trans. Robot.,* http://nxr.northwestern.edu/publications, In Review.

[3] A. Mavrommati, A. Ansari, and T. Murphey, "Optimal control-on-request: An application in real-time assistive balance control," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2015, pp. 5928–5934.

[4] M. da Silva, Y. Abe, and J. Popović, "Interactive simulation of stylized human locomotion," *ACM Transactions on Graphics (TOG)*, vol. 27, no. 3, p. 82, 2008.

[5] Y. Tassa, T. Erez, and W. D. Smart, "Receding horizon differential dynamic programming," in *Advances in neural information processing systems*, 2008, pp. 1465–1472.

[6] Y. Ye and C. K. Liu, "Optimal feedback control for character animation using an abstract model," in *SIGGRAPH 2010*, vol. 29, no. 4. ACM, 2010, p. 74.

[7] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 4906–4913.

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

[9] D. Baraff, "Fast contact force computation for nonpenetrating rigid bodies," in *Proceedings of the 21st Annual Conference on Computer graphics and Interactive Techniques*. ACM, 1994, pp. 23–34.

[10] B. Mirtich and J. Canny, "Impulse-based simulation of rigid bodies," in *Proceedings of the 1995 Symposium on Interactive 3D graphics*. ACM, 1995, pp. 181–189.

[11] Y. Tassa and E. Todorov, "Stochastic complementarity for local control of discontinuous dynamics," in *Robotics: Science and Systems*. Citeseer, 2010.

[12] E. Todorov, "Implicit nonlinear complementarity: A new approach to contact dynamics," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 2322–2329.

[13] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 1168–1175.

[14] H. Geyer, A. Seyfarth, and R. Blickhan, "Compliant leg behaviour explains basic dynamics of walking and running," *Proceedings of the Royal Society B: Biological Sciences*, vol. 273, no. 1603, pp. 2861–2867, 2006.

[15] P. Holmes, R. J. Full, D. Koditschek, and J. Guckenheimer, "The dynamics of legged locomotion: Models, analyses, and challenges," *Siam Review*, vol. 48, no. 2, pp. 207–304, 2006.

[16] M. Hutter, C. Remy, M. Hoepflinger, and R. Siegwart, "Scarleth: Design and control of a planar running robot," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011, pp. 562–567.

[17] M. Ahmadi and M. Buehler, "Controlled passive dynamic running experiments with the ARL-monopod II," *IEEE Trans. on Robotics*, vol. 22, no. 5, pp. 974–986, 2006.

[18] J. W. Hurst, J. E. Chestnutt, and A. A. Rizzi, "The actuator with mechanically adjustable series compliance," *IEEE Trans. on Robotics*, vol. 26, no. 4, pp. 597–606, 2010.

[19] M. H. Raibert, "Legged robots," *Communications of the ACM*, vol. 29, no. 6, pp. 499–514, 1986.

[20] W. J. Schwind and D. E. Koditschek, "Control of forward velocity for a simplified planar hopping robot," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, 1995, pp. 691–696.

[21] J. Pratt, C.-M. Chew, A. Torres, P. Dilworth, and G. Pratt, "Virtual model control: An intuitive approach for bipedal locomotion," *The Int. Journal of Robotics Research*, vol. 20, no. 2, pp. 129–143, 2001.

[22] M. M. Ankaralı and U. Saranlı, "Analysis and control of a dissipative spring-mass hopper with torque actuation," *Robotics: Science and Systems*, vol. 4, no. 6, p. 8, 2010.

[23] D. Koepl and J. Hurst, "Impulse control for planar spring-mass running," *Journal of Intelligent & Robotic Systems*, vol. 74, no. 3-4, pp. 589–603, 2014.

[24] H. Geyer, A. Seyfarth, and R. Blickhan, "Spring-mass running: simple approximate solution and application to gait stability," *Journal of theoretical biology*, vol. 232, no. 3, pp. 315–328, 2005.

[25] M. Hutter, C. D. Remy, M. Hopflinger, and R. Siegwart, "Slip running with an articulated robotic leg," in *IEEE Int./RSJ Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2010, pp. 4934–4939.

[26] M. Raibert, K. Blankespoor, G. Nelson, R. Playter, *et al.*, "Bigdog, the rough-terrain quadruped robot," in *Proceedings of the 17th World Congress*, 2008, pp. 10 823–10 825.

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

[28] A. Ansari, K. Flaßkamp, and T. D. Murphey, "Sequential action control for tracking of free invariant manifolds," in *Proc. Conf. Anal. Des. Hybrid Sys. (ADHS)*, 2015, pp. 335–342.

[29] M. Egerstedt, Y. Wardi, and H. Axelsson, "Transition-time optimization for switched-mode dynamical systems," *IEEE Transactions on Automatic Control*, vol. 51, no. 1, pp. 110–115, 2006.

[30] H. Chen and F. Allgöwer, "Nonlinear model predictive control schemes with guaranteed stability," in *Nonlinear model based process control*. Springer, 1998, pp. 465–494.

[31] 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, 2000.

[32] L. Grüne and J. Pannek, *Nonlinear model predictive control*. Springer, 2011.

[33] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.

[34] H. Chen and F. Allgower, "A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability," in *European Control Conference (ECC)*, 1997, pp. 1421–1426.

[35] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[36] L. T. Biegler, "A survey on sensitivity-based nonlinear model predictive control," in *International Symposium on Dynamics and Control of Process Systems (IFAC)*, 2013, pp. 499–510.

[37] B. D. Anderson and J. B. Moore, *Optimal control: linear quadratic methods*. Courier Corporation, 2007.

[38] O. Arslan, "Model based methods for the control and planning of running robots," Ph.D. dissertation, Bilkent University, 2009.

[39] E. Johnson, J. Schultz, and T. Murphey, "Structured linearization of discrete mechanical systems for analysis and optimal control," *IEEE Trans. on Automation Science and Engineering*, vol. 12, no. 1, pp. 140–152, 2014.

[40] I. Poulakakis and J. W. Grizzle, "The spring loaded inverted pendulum as the hybrid zero dynamics of an asymmetric hopper," *IEEE Trans. on Automatic Control*, vol. 54, no. 8, pp. 1779–1793, 2009.

[41] A. D. Wilson, J. Schultz, A. Ansari, and T. D. Murphey, "Real-time trajectory synthesis for information maximization using sequential action control and least-squares estimation," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2015, pp. 4935–4940.

[42] P. A. Parrilo, "Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization," Ph.D. dissertation, California Institute of Technology, 2000.