Controllers as Filters: Noise-Driven Swing-Up Control Based on Maxwell's Demon

Emmanouil Tzorakoleftherakis, *Student Member, IEEE* and Todd D. Murphey, *Member, IEEE*

***Abstract*— In this paper we show in simulation that if the controller and the filter are combined into a single computational unit operating like a "Maxwell's demon", controller-filtered noise can successfully control a system of interest. Using this method, we perform Monte Carlo tests for the swing-up control of the cart pendulum, acrobot and pendubot systems. Results show that filtered noise can indeed act as a swing-up controller, leading these systems to configurations where a locally stabilizing controller can complete the stabilization process. Potential applications of this work include the development of reliable, never-failing human-machine interfaces for rehabilitation, training and skill evaluation.**

I. INTRODUCTION

All human-machine interfaces involve control signals with substantial uncertainty ([1]–[4]). Using the part of the signal that contributes to the control objective while ignoring the part of the signal that does not, is important to the interface design. In this paper, we present a control algorithm that addresses this issue, and can be extended for utility in human-machine interface synthesis. The algorithm is based on Maxwell's demon, a character introduced by Maxwell in 1871 [5] to contradict the second law of thermodynamics, which asserts the irreversibility of natural processes, or, as stated in Clausius's version: *"It is impossible to devise an engine which, working in a cycle, shall produce no effect other than the transfer of heat from a colder to a hotter body"* [6].

Maxwell's demon operates on a box filled with a gas. Suppose that a partition is placed across the middle of the box such that the left and right side have equal volume. The gas was initially in thermal equilibrium at some temperature, and thus the same temperature is retained in both sides after the partition is placed, meaning that the gas molecules are moving at a certain average velocity. The demon observes the molecules on both sides of the box; if he sees one approaching that is moving slower than the average, he allows it through a molecule-sized door to the right side of the box. Similarly, if a molecule approaches the door with a speed greater than the average, it ends up on the left side of the box. After the demon's work is complete, the left and right sides of the box are filled with hot and cold gas respectively. Then, unlike what the second law states, one could use this temperature difference to run a heat engine by allowing the heat to flow from the hot side to the cold side. Although several physicists showed that a more complete analysis of the whole system, including the demon, does not violate the second law of thermodynamics [7], the demon conceptually survived until 1961 when Landauer put an end to his life [8] by introducing a new concept of "logical irreversibility". Still, this paradox has contributed to a new paradigm over the past century, i.e. the interaction of physics and information theory.

The method proposed here is similar to Maxwell's demon, filtering out parts of a potential input signal that do not contribute to the control objective. At every moment, our Maxwell's demon "looks" at the nominal control input (controller) to determine whether or not to pass an external signal to the system as an input (filter). Thus, the controller and the filter are combined into a single computational unit, i.e. a "demon". We show in simulation that *noise* filtered by this approach can control a system of interest. To decide whether a specific noise sample will be accepted or not, we use the inner product between the control and noise vectors as a metric. If the inner product is positive and the angle between the two vectors is small enough, the noise vector should move the system towards the desired control direction. The controller/filter we use is a recently formulated algorithm for control of nonlinear systems called sequential action control (SAC) presented in [9], [10], [11] and summarized in Section II. Nevertheless, it must be noted that the implementation of this method is not SAC-specific; any controller that can successfully control the system of interest can be used instead.

As a test platform, we use the swing-up problem of the cart-pendulum, pendubot and acrobot systems ([9], [12]–[18]), benchmark examples for nonlinear control methods and techniques. The objective of swing-up control is to swing the system to a small neighborhood $\mathcal{W}_s$ of the upright unstable equilibrium, where, often, control authority is ceded to a locally stabilizing controller, e.g. a linear quadratic regulator (LQR), that stabilizes the system about the upward vertical position.

The contribution of this paper is two-fold. First, we illustrate an interesting application of the Maxwell's demon philosophical idea to swing-up control. Secondly, and more importantly, we provide evidence that noisy inputs can be a rich source of control authority, which has importance particularly in rehabilitation and assistive technologies where an impaired subject may introduce substantial amounts of noise and structured uncertainty into the control inputs ([1]–[4]).

The rest of the paper is structured as follows: in Section II we provide a description of the nonlinear controller that is

Authors are with the Neuroscience and Robotics Laboratory (N×R) at the Department of Mechanical Engineering, Northwestern University, Evanston, IL. Email: man7therakis@u.northwestern.edu, t-murphey@northwestern.edu

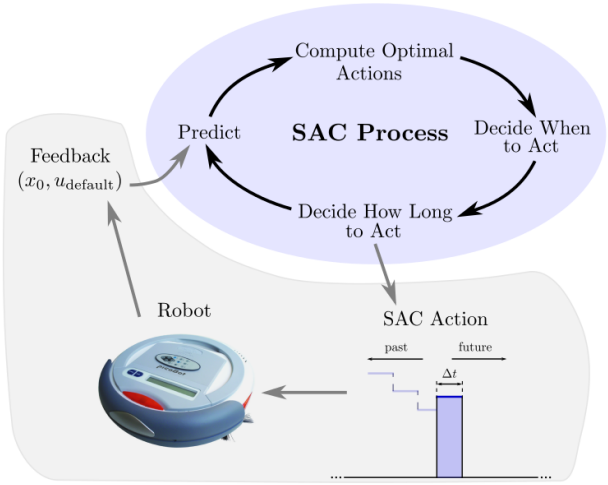

Fig. 1. An overview of the SAC control process.

used as a filter in this paper, followed by the rationale behind noise-driven control explained in Section III. Section IV demonstrates in simulation how the proposed method can successfully swing up the inverted pendulum, acrobot and pendubot systems initialized at the downward stable equilibrium. Finally, Section V discusses potential applications of this method and future work directions.

II. Preliminaries - Sequential Action Control

In this paper we use sequential action control (SAC), a recently formulated algorithm for control of nonlinear systems, as a *filter* that rejects noise samples not driving the system to the desired control direction. For convenience, we will now briefly summarize the work presented in [9]–[11].

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

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

As opposed to alternative 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 incremental cost $l(x(t))$ and terminal cost $m(x(t_0 + T))$.[1] As a result of its control synthesis process, SAC can calculate controls that optimally improve (2) even in the cases 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 are described in the subsequent subsections.

A. 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 $u = u_{default}$. The horizon length T, is a design parameter. Without loss of generality, the default control throughout this paper is null, $u_{default} \triangleq 0$. The term is included in formulas for completeness and indicates potential shared control implementation.[2]

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

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

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).

B. 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-C) and the control duration (Section II-D). 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

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

in driving expected change in cost (sensitivity $\frac{dJ_{track}}{d\lambda}$) to a user specified design parameter $\alpha_d \in \mathbb{R}^-$. This parameter 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-A), the control schedule that minimizes (4) is provided as a closed-form expression,

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

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

[1] Although (2) lacks a norm on control effort, SAC includes this norm in the following step, in (4).

[2] As an example, $u_{default}$ may be an optimized feedforward controller providing a nominal trajectory around which SAC would provide feedback.

C. Decide When to Act

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 that optimizes the trade-off between the cost of waiting and the efficacy of control at that time according to,

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

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

D. Decide How Long to Act

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 [19]. 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 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 $u_{default} = 0$, 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 [9] for proof). By iterating on this process (Section II-A until Section II-D), SAC rapidly synthesizes piecewise-continuous, constrained control laws for nonlinear systems. For more information about SAC, the reader is encouraged to consult [9]–[11].

III. Noise-driven Control Based on Maxwell's Demon

This section describes the approach behind noise-driven control. Our Maxwell's demon can be implemented by combining a controller and a filter into a single computational unit that filters out noise not driving the system to the desired control direction.

Assume that at sampling intervals, noise enters a system, i.e. a sample noise vector $\epsilon \in \mathbb{R}^m$ is drawn from a distribution every t_s seconds. Further, suppose that at each time instance a vector of potential control inputs is computed based on a controller (in this paper we use SAC). Note that the controller should be capable of driving the system by itself according to the desired specifications. If the inner product between the control and the noise vector is positive, and the corresponding angle of the vectors is small, then the effect of noise on the system should be similar to that of the control vector. In that case, noise is allowed to pass through the filter/controller. If any of the above two conditions is violated, noise is "rejected" (i.e. no input applied to the system). In theory, noise filtered by this approach should drive the system towards the desired control direction; in case the "Maxwell's demon" is based on an optimal controller in particular, these directions will improve the objective. This process is illustrated in Algorithm 1. In the following Section, we use Algorithm 1 in swing-up control and provide evidence that noisy inputs can be a rich source of control authority.

Algorithm 1 Noise-driven control using Maxwell's demon

- Select noise distribution, mean μ, standard deviation σ and associated parameters accordingly.
- Initialize current time t_0, sampling time t_s, final time t_f, angle tolerance γ.

while $t_0 < t_f$ **do**
  Sample ϵ from noise distribution
  Compute controller value u
  Calculate inner product $\langle u, \epsilon \rangle$
  Calculate angle ϕ between u and ϵ
  if $\langle u, \epsilon \rangle > 0$ **and** $|\phi| \leq \gamma$
    Use ϵ as current input, $u_{curr} = \epsilon$
  else
    Completely "reject" ϵ, $u_{curr} = 0$
  end if
  Apply u_{curr} for $t \in [t_0, t_0 + t_s]$
  $t_0 = t_0 + t_s$
end while

IV. Simulation Results

This section applies the proposed algorithm in simulation examples where SAC-filtered *Gaussian* noise is used as a control input on-line aiming to swing up three benchmark systems, i.e. the cart pendulum, the acrobot and the pendubot. Once the system is within a specified region $\mathcal{W}_s$, the control switches from SAC-filtered noise to an LQR that completes the stabilization process about the upward vertical equilibrium (region $\mathcal{W}_o$). Regions $\mathcal{W}_s$ and $\mathcal{W}_o$ are selected in the following paragraphs.

For each of the three systems, we performed a Monte Carlo test of 500 simulations. For all simulations, $t_0 = 0s$ and $t_f = 25s$; if noise could not swing up the system within 25 seconds, i.e. $x(25) \notin \mathcal{W}_s$, the trial was considered failed. Similarly there were some cases when $\mathcal{W}_s$ was reached right before the $25s$ limit not leaving enough time to the LQR for stabilization, i.e. to reach $\mathcal{W}_o$. These cases were noted but they are not indicative of the algorithm's performance. Because there is only one control input in these systems, the control and noise vectors were always parallel and their

relative angle, ϕ was either $180°$ or $0°$; thus the parameter γ of Algorithm 1 was set to 0. Moreover, for the SAC objective (2) we used $l(x(t)) = \frac{1}{2}x(t)^T Q\, x(t)$ – a typical quadratic cost – and $m(x(t_0+T)) = \frac{1}{2}x(t_0+T)^T P\, x(t_0+T)$. The values of Q, P and T used in the simulations are shown in Table II.

To assess the algorithm's performance, in each trial we recorded the switching time, i.e. the time the system reached $\mathcal{W}_s$, and the stabilization time, i.e. the time $\mathcal{W}_o$ was reached. Furthermore, to evaluate the resulting trajectories, we defined a trajectory cost J as:

$$J = \int_0^{25} \frac{1}{2} x(t)^T Q_1\, x(t)\, dt, \text{ with } Q_1 = \mathbb{I}^{4\times 4}. \quad (7)$$

For consistency, this metric was calculated only for trajectories that reached both $\mathcal{W}_s$ and $\mathcal{W}_o$. Finally, the resulting histograms of the switching time, stabilization time and cost J were compared with their corresponding nominal values, that were calculated by applying the actual SAC output to the system (instead of noise).

A. Cart-Pendulum System

The equations that describe the underactuated cart-pendulum system [17], [18] (see Fig. 2a) are given by:

$$\dot{x} = f(x,u) = \begin{pmatrix} \dot{\theta} \\ \frac{g}{l}\sin\theta + u\cos\theta - \frac{b}{ml^2}\dot{\theta} \\ \dot{x}_c \\ u \end{pmatrix} \quad (8)$$

TABLE I
PARAMETER VALUES FOR SYSTEM DYNAMICS

Cart pendulum (Fig. 2a)		Fig. 2b	Acrobot	Pendubot
Dynamics	Value	**Dynamics**	Value	Value
m	0.2 kg	m_1	1 kg	1 kg
l	1 m	l_1	1 m	2 m
b	0.01 Ns/m	m_2	0.5 kg	0.5 kg
g	9.81 m/s^2	l_2	2 m	1 m
		g	9.81 m/s^2	9.81 m/s^2

TABLE II
PARAMETER VALUES FOR SAC AND ALGORITHM 1

SAC	Cart Pendulum	Acrobot	Pendubot
Q from (2)	$[200, 0, 100, 50]^*$	$[1000, 30, 0, 0]^*$	$[1000, 30, 0, 0]^*$
P from (2)	0	$[50, 150, 0, 0]^*$	$[50, 200, 0, 0]^*$
R from (4)	0.3	0.1	0.1
α_d	$-5J_{track}$	$-5J_{track}$	$-5J_{track}$
T	$1.2s$	$1.2s$	$1.2s$
Algorithm 1	Cart Pendulum	Acrobot	Pendubot
t_0	$0s$	$0s$	$0s$
t_f	$25s$	$25s$	$25s$
t_s	$0.02s$	$0.0025s$	$0.0025s$
γ	0	0	0
μ	0	0	0
σ	10	18	40

* Diagonal matrix.

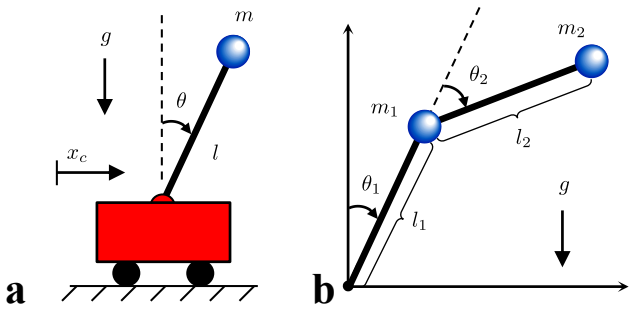

Fig. 2. Schematic of the cart pendulum system (a) and the acrobot/pendubot (b).

where the state vector consists of the angular position and velocity of the pendulum and the position and lateral velocity of the cart, $x = [\theta\ \dot{\theta}\ x_c\ \dot{x}_c]^T \in \mathcal{X}$, the input u is the lateral acceleration of the cart, g is the acceleration due to gravity, b is the damping coefficient, l is the pendulum length and m the mass at the tip, as shown in Fig. 2a.

The parameters of the dynamics, SAC and Algorithm 1 that were used in the simulations are given in Tables I and II. The regions of interest $\mathcal{W}_s$ and $\mathcal{W}_o$ were selected as: [3]

$$\mathcal{W}_s = \left\{ x \in \mathcal{X} :\ |\theta| < 0.8\, rad\ \wedge\ |\dot{\theta}| < 1\, rad/s \right\} \quad (9)$$

$$\mathcal{W}_o = \{ x \in \mathcal{X} : \wedge\ |x_i| < 0.02,\ \ i = 1,2,3,4 \} \quad (10)$$

and the LQR gains calculated offline about the inverted unstable equilibrium for final stabilization are:

$$K = \begin{pmatrix} 52.76 & 16.98 & -3.16 & -6.14 \end{pmatrix}. \quad (11)$$

Figure 3 shows the result of a sample simulation. By observing the first two plots, the resemblance between Maxwell's demon and our controller/filter is obvious; SAC rejects noise samples that would otherwise lead the system to a different direction.

The results of the Monte Carlo test are shown in the first column of Fig. 4. It can be seen that SAC-filtered noise was able to successfully lead the system to the switching region $\mathcal{W}_s$ in all 500 trials (0% failure rate). Additionally, the histogram ranges of the switching time, stabilization time and metric J were similar to their nominal values, that were calculated by applying the actual SAC output to the system (instead of noise).

B. Acrobot

For the two-link planar robot shown in Fig. 2b, the simplified dynamics in the absence of friction, assuming point masses and a torque input at the elbow joint ([13], [14]) are given by:

$$M(\theta)\ddot{\theta} + H(\theta, \dot{\theta}) + G(\theta) = Bu \quad (12)$$

[3] In (10), x_i is the i_{th} element of the state vector x.

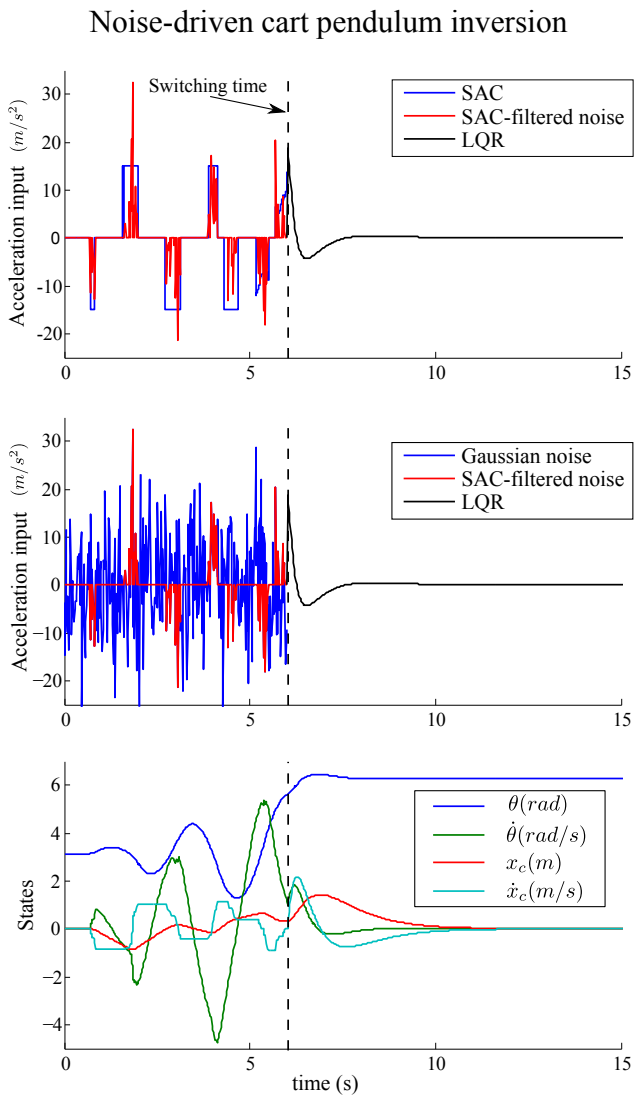

Fig. 3. Sample simulation for the cart pendulum system. The top two plots demonstrate the similarity of our controller/filter with Maxwell's demon; SAC as a filter rejects noise samples that would otherwise lead the system to a different control direction.

where u is the torque input, $\theta = [\theta_1 \ \theta_2]^T$,

$$M(\theta) = \begin{pmatrix} M_{1,1} & M_{1,2} \\ M_{2,1} & M_{2,2} \end{pmatrix} = \begin{pmatrix} \alpha_1 + \alpha_2 + 2\alpha_3 \cos\theta_2 & \alpha_3 \cos\theta_2 + \alpha_2 \\ \alpha_3 \cos\theta_2 + \alpha_2 & \alpha_2 \end{pmatrix} \quad (13)$$

$$H(\theta, \dot{\theta}) = \begin{pmatrix} H_1 \\ H_2 \end{pmatrix} = \begin{pmatrix} -2\alpha_3 \dot{\theta}_1 \dot{\theta}_2 \sin\theta_2 - \alpha_3 \dot{\theta}_2^2 \sin\theta_2 \\ \alpha_3 \dot{\theta}_1^2 \sin\theta_2 \end{pmatrix} \quad (14)$$

$$G(\theta) = \begin{pmatrix} G_1 \\ G_2 \end{pmatrix} = \begin{pmatrix} -\beta_1 \sin\theta_1 - \beta_2 \sin(\theta_1 + \theta_2) \\ -\beta_2 \sin(\theta_1 + \theta_2) \end{pmatrix} \quad (15)$$

$$B = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad (16)$$

and

$$\alpha_1 = (m_1 + m_2) l_1^2, \ \ \alpha_2 = m_2 l_2^2, \ \ \alpha_3 = m_2 l_1 l_2$$
$$\beta_1 = (m_1 + m_2) g l_1, \ \ \beta_2 = m_2 g l_2. \quad (17)$$

Thus, we can rewrite (12) as:

$$\dot{x} = f(x, u) = \begin{pmatrix} \dot{\theta} \\ -M^{-1}(H + G) + M^{-1} B u \end{pmatrix} \quad (18)$$

with $x = [\theta \ \dot{\theta}]^T \in \mathcal{X}$. The rest of the parameters are seen in Fig. 2b and their values, along with the parameters of SAC and Algorithm 1 that were used in the Monte Carlo simulations are given in Tables I and II. The regions of interest were selected as:

$$\mathcal{W}_s = \Big\{ x \in \mathcal{X} : \ |\theta_{1,2}| < 0.8\, rad \ \wedge \ |\dot{\theta}_{1,2}| < 2\, rad/s \ \wedge \ \theta_1 \theta_2 < 0 \Big\} \quad (19)$$

and $\mathcal{W}_o$ was the same as in (10). The corresponding LQR gains for final stabilization are:

$$K = \begin{pmatrix} -257.13 & -120.1 & -103.77 & -59.22 \end{pmatrix}. \quad (20)$$

The results of the Monte Carlo test are shown in the second column of Fig. 4. Unlike the cart pendulum case, there was 8% probability of swing-up failure, i.e. noise did not lead the system to $\mathcal{W}_s$ 40 out of the total 500 trials. Even though this probability is relatively low, it is possible that it can be reduced further by selecting a finer estimate of the switching region $\mathcal{W}_s$. In other words, given that all trajectories that reached $\mathcal{W}_s$ eventually reached $\mathcal{W}_o$ as well, the region $\mathcal{W}_s$ is most likely underestimating the LQR region of attraction. Finally, regarding the nominal values calculated using the actual SAC output (red dashed lines), it can be seen that they were close to the corresponding histogram ranges as expected.

C. Pendubot

The dynamics of the pendubot ([15], [16]) are also given by (12)-(18), with $B = (1 \ \ 0)^T$, i.e. the torque input is applied at the shoulder joint. The parameters of the dynamics, SAC and Algorithm 1 used in the simulations are given in Tables I and II.

$\mathcal{W}_o$ was the same as in (10) while $\mathcal{W}_s$ was:

$$\mathcal{W}_s = \Big\{ x \in \mathcal{X} : \ |\theta_{1,2}| < 0.35\, rad \ \wedge \ |\dot{\theta}_1| < 0.45\, rad/s \ \wedge \ |\dot{\theta}_2| < 0.75\, rad/s \ \wedge \ \theta_1 \theta_2 < 0 \Big\}. \quad (21)$$

Upon linearization about the origin, the LQR gains we obtained are the following:

$$K = \begin{pmatrix} -156.66 & -90.18 & -156.62 & -46.78 \end{pmatrix}. \quad (22)$$

The results of the Monte Carlo test are shown in the third column of Fig. 4. There was only 1% probability of failure to reach $\mathcal{W}_s$. However, because the switching time histogram had high variance (higher than the two previous cases), 5% of the trials that reached $\mathcal{W}_s$ within $25s$ (25 out of 495), failed to reach $\mathcal{W}_o$ as well. Specifically, a significant amount of trials entered $\mathcal{W}_s$ when $t > 20s$, and as a result, the LQR

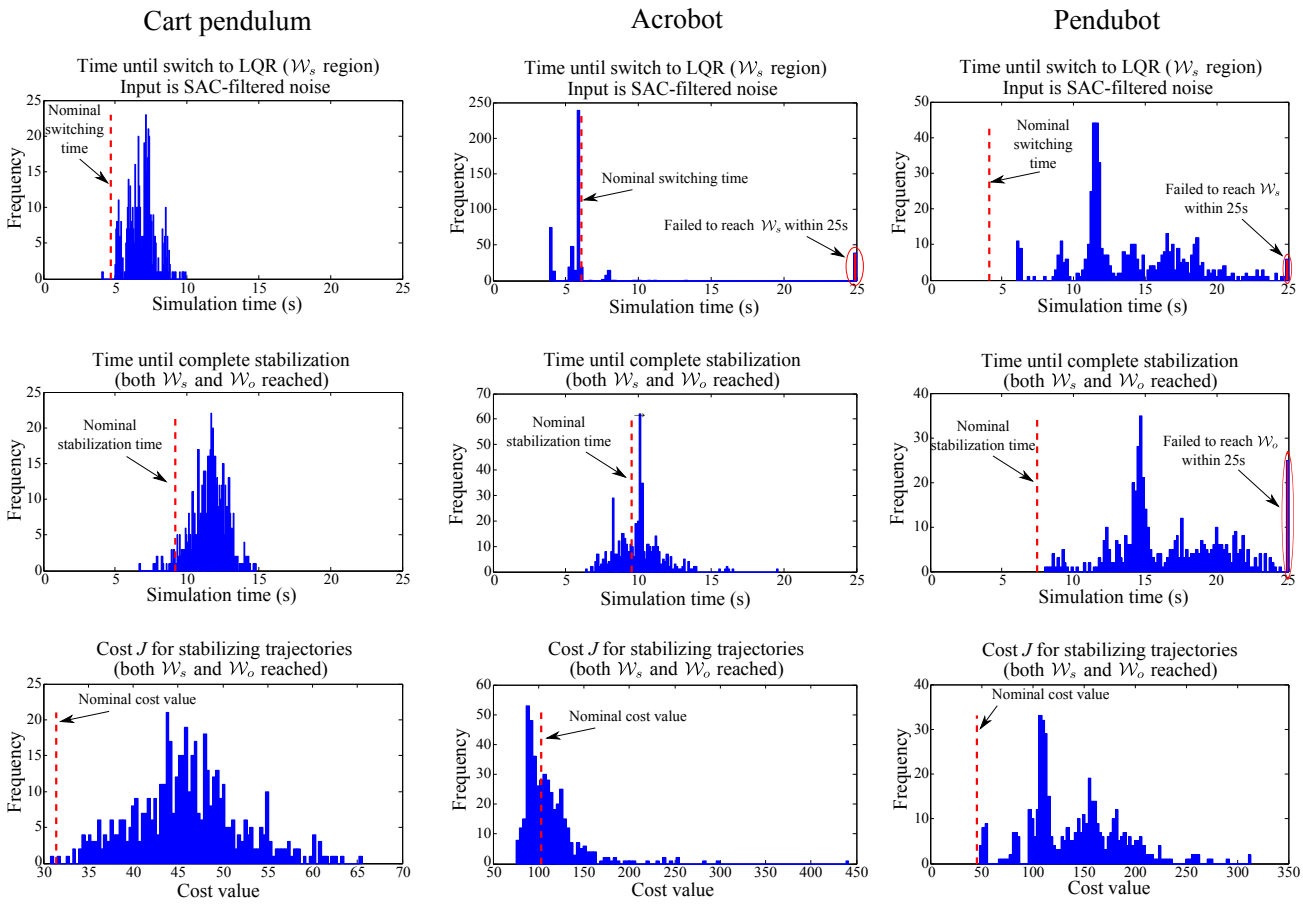

Fig. 4. Results of Monte Carlo simulations; each column corresponds to a different system. The nominal values of switching time, stabilization time and cost J (red dashed lines) are calculated by applying the actual SAC output to the system (instead of noise). For the pendulum system, the swing-up control was successful in all 500 iterations. For the acrobot, there was 8% probability of failure. Since all trajectories that reached $\mathcal{W}_s$ eventually reached $\mathcal{W}_o$ as well, a potential explanation is that $\mathcal{W}_s$ is underestimating the LQR region of attraction. Hence, for larger $\mathcal{W}_s$, this probability will most likely be lower. Finally, for the pendubot there was 1% probability of failure to reach $\mathcal{W}_s$. Additionally, 5% of the trials that reached $\mathcal{W}_s$ within $25s$ (25 out of 495), failed to reach $\mathcal{W}_o$ as well. This can be explained by the high variance observed in the switching time histogram. A significant amount of trials entered $\mathcal{W}_s$ when $t > 20s$, and as a result, the LQR did not have enough time to lead the system to $\mathcal{W}_o$. This is an artifact of the $25s$ time limit and is not indicative of the algorithm's performance; for larger t_f values, the number of trials failing to reach $\mathcal{W}_s$ and $\mathcal{W}_o$ would be expected to drop.

did not have enough time to lead the system to $\mathcal{W}_o$. This is an artifact of the $25s$ time limit and is not indicative of the algorithm's performance; for larger t_f values, the number of trials failing to reach $\mathcal{W}_s$ and $\mathcal{W}_o$ would be expected to drop. Finally, due to the high variance observed, the nominal values were – within reason – smaller than the average histogram value.

V. Discussion and Concluding Remarks

In this paper, we presented an algorithm that applies the Maxwell's demon philosophical idea to swing-up control. At every moment, our Maxwell's demon "looks" at a control input and determines whether or not to pass a signal to the control system as an input. The algorithm can work with any controller that can successfully control the system of interest. Additionally, we showed in simulation that, if filtered with Algorithm 1, noisy inputs can be a useful source of control inputs, even for demanding swing-up tasks.

The effect of the noise distribution and standard deviation σ (or signal-to-noise ratio) was not considered and will be examined in our future work. However, as a rule of thumb, the value of σ should be comparable to the value of the nominal controller's output (in this case, SAC). Moreover, Algorithm 1 can be easily modified to account for higher σ values, by adding a saturation step for the noisy input after the inner product test. The sampling time, t_s, is another parameter that can affect the algorithm's performance. In general, the smaller the sampling time, the more reliable the algorithm. Furthermore, for the swing-up problem, if t_s is small enough, noise can completely stabilize the system without switching to a locally stabilizing controller. This is demonstrated for the simulated cart pendulum system online at https://vimeo.com/nxrlab/maxdemon. Here, SAC was initially used as the controller/filter, and when the system reached $\mathcal{W}_s$, potential noise inputs were filtered by an LQR.

Although noise cannot be used in practice to directly control a real system because it requires motors to generate non-smooth signals, this result can be used towards the development of reliable *software* interfaces for human-

machine interactions. In our future work, we will examine the feasibility of our method as an interface in a human subject study. Potential fields of application include human training [20], skill evaluation and most importantly, rehabilitation and assistive technologies where sensory-motor deficits can result in involuntary tremor and spasticity, which introduce substantial amounts of noise and structured uncertainty into the control inputs([1]–[4]).

In a practical application of Algorithm 1 as a human-machine interface, ϵ represents the *user input*. In this aspect, the user input will have either a positive or a negative inner product with the control vector u. To minimize threat to the user and also to the system, that originates from novice or involuntary actions, if the inner product is negative, the user's input is "rejected" and the system input is set equal either to zero or to the nominal control value. In the latter case, this will result in an interface that potentially *never fails*, serving both training and safety purposes. Additionally, the percentage of "accepted" user actions (PAA) can be used as a metric that evaluates expertise or even training progress. For instance, in case of robot-assisted stroke rehabilitation [3], the progress of stroke survivors can be monitored by recording the PAA. Similarly, Algorithm 1 can be applied to lower extremity exoskeletons, to filter out spastic movements after spinal cord injury.

Lastly, as mentioned in Section II, SAC can calculate controls that optimally improve (2) even in the cases where the objective is non-convex or unbounded. Hence, if there are multiple objectives defined for one system, each one associated with a separate SAC process, Algorithm 1 can be used to tie a user's input to the nearest SAC controller, and thus objective. Consequently, the controller/filter will have a dual role; it will estimate what the user intent is as well as only allow acceptable control actions to be filtered through to the system.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant CNS 1329891. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] B.-C. Chen and H. Peng, "Differential-braking-based rollover prevention for sport utility vehicles with human-in-the-loop evaluations," *Vehicle System Dynamics*, vol. 36, no. 4-5, pp. 359–389, 2001.

[2] J. Tang, Q. Zhao, and R. Yang, "Stability control for a walking-chair robot with human in the loop," *International Journal of Advanced Robotic Systems*, vol. 6, no. 1, pp. 47–52, 2009.

[3] H. Kazerooni, J.-L. Racine, L. Huang, and R. Steger, "On the control of the berkeley lower extremity exoskeleton (BLEEX)," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005, pp. 4353–4360.

[4] M.-S. Ju, C.-C. Lin, D.-H. Lin, I.-S. Hwang, and S.-M. Chen, "A rehabilitation robot with force-position hybrid fuzzy controller: hybrid fuzzy control of rehabilitation robot," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 13, no. 3, pp. 349–358, 2005.

[5] H. Leff and A. F. Rex, *Maxwell's Demon 2 Entropy, Classical and Quantum Information, Computing*. CRC Press, 2002.

[6] A. B. Pippard, *Elements of classical thermodynamics: for advanced students of physics*. Cambridge University Press, 1957.

[7] T. Sagawa, "Thermodynamics of information processing in small systems," *Progress of Theoretical Physics*, vol. 127, no. 1, pp. 1–56, 2012.

[8] R. Landauer, "Irreversibility and heat generation in the computing process," *IBM Journal of Research and Development*, vol. 5, no. 3, pp. 183–191, 1961.

[9] A. R. Ansari and T. D. Murphey, "Sequential action control: Closed-form optimal control for nonlinear systems," *IEEE Transactions on Robotics*. In Review. http://nxr.northwestern.edu/publications, In Review.

[10] A. Mavrommati, A. Ansari, and T. Murphey, "Optimal control-on-request: An application in real-time assistive balance control," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Accepted. http://nxr.northwestern.edu/publications, 2015.

[11] A. R. Ansari and T. D. Murphey, "Control-on-request: Short-burst assistive control for long time horizon improvement," in *American Control Conference (ACC)*. Accepted. http://nxr.northwestern.edu/publications, 2015.

[12] I. Fantoni, R. Lozano, and M. W. Spong, "Energy based control of the pendubot," *IEEE Transactions on Automatic Control*, vol. 45, no. 4, pp. 725–729, 2000.

[13] X. Xin and T. Yamasaki, "Energy-based swing-up control for a remotely driven acrobot: Theoretical and experimental results," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 4, pp. 1048–1056, 2012.

[14] M. W. Spong, "The swing up control problem for the acrobot," *IEEE Control Systems*, vol. 15, no. 1, pp. 49–55, 1995.

[15] 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 *Proceedings of the IEEE Conference on Decision and Control (CDC)*, 2006, pp. 6138–6143.

[16] M. W. Spong and D. J. Block, "The pendubot: A mechatronic system for control research and education," in *Proceedings of the IEEE Conference on Decision and Control (CDC)*, vol. 1, 1995, pp. 555–556.

[17] C.-W. Tao, J.-S. Taur, T. W. Hsieh, and C. Tsai, "Design of a fuzzy controller with fuzzy swing-up and parallel distributed pole assignment schemes for an inverted pendulum and cart system," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 6, pp. 1277–1288, 2008.

[18] K. J. Åström and K. Furuta, "Swinging up a pendulum by energy control," *IFAC, San Francisco*, vol. 13, 1996.

[19] T. M. Caldwell and T. D. Murphey, "Projection-based switched system optimization: Absolute continuity of the line search," in *Proceedings of the IEEE Conference on Decision and Control (CDC)*, 2012, pp. 699–706.

[20] E. Tzorakoleftherakis, F. A. Mussa-Ivaldi, R. A. Scheidt, and T. D. Murphey, "Effects of optimal tactile feedback in balancing tasks: A pilot study," in *American Control Conference (ACC)*, 2014, pp. 778–783.