Discrete-Decision Continuous-Actuation control: balance of an inverted pendulum and pumping a pendulum swing

Pranav A. Bhounsule[1], **Andy Ruina**[2] **and Gregg Stiesberg**[3]
[1] Mechanical Engineering, University of Texas at San Antonio, One UTSA Circle, San Antonio, TX, 78249
[2] Mechanical Engineering, Cornell University, Ithaca, NY, 14853
[3] Physics; Cornell University, Ithaca, NY 14853
emails: pranav.bhounsule@utsa.edu, ruina@cornell.edu, grs26@cornell.edu

J. Dyn. Sys., Meas., Control 137(5), 051012 (2015) (9 pages); Paper No: DS-14-1118; doi:10.1115/1.4028851

In some practical control problems of essentially-continuous systems, the goal is not to tightly track a trajectory in state space, but only some aspects of the state at various points along the trajectory, and possibly only loosely. Here we show examples in which classical discrete-control approaches can provide simple, low input- and low output- bandwidth control of such systems. The sensing occurs at discrete state- or time-based events. Based on the state at the event, we set a small set of control parameters. These parameters prescribe features, e.g. amplitudes of open-loop commands that, assuming perfect modeling, force the system to, or towards, goal points in the trajectory. Using this discrete decision continuous actuation (DDCA) control approach, we demonstrate stabilization of two examples: **1)** linear dead-beat control of a time delayed linearized inverted pendulum; and **2)** pumping of a hanging pendulum. Advantages of this approach include: It is computationally cheap compared to real-time control or online optimization; it can handle long time delays; it can fully correct disturbances in finite time (dead-beat control); it can be simple, using few control gains and set points and limited sensing; and it is low bandwidth for both sensing and actuator commands. We have found the approach useful for control of robotic walking.

1 Introduction

While digital controllers are by nature discrete and not continuous, the basic nature of discrete controllers is not well known by many. This paper, and the demonstrations described here, were prepared because our robotic control approach was criticized at a meeting with the following claims: "It is well known that a linear control cannot be dead-beat, and it is well known that a proportional controller cannot control a system with time delays bigger than the characteristic instability times of the uncontrolled plant." Both of these claims are false for discrete control. And discrete control has some other advantages, even for continuous systems.

We do not claim that any particular result here is original, or even surprising to appropriate experts. Rather, we want to lay out simply some simple, but under-appreciated by some, discrete control ideas.

Sometimes good-enough control need only get some aspects of the system state to occasionally nearly match a goal. For example, for control of walking, good balance might be achieved if, about once per step, the body is moving at about the right speed without having failed (fallen). That is, in the whole time-trajectory of the robot parts, only a few key goal points need be achieved, and then maybe only approximately, in each step. The details of the rest of the motion are unimportant. So why not ignore them? Given that the goal is essentially low dimensional, why should a controller be complex or use many numbers to sense the state or to command the actuators (e.g., why use a fast stream of input and output commands)? It is interesting to seek a control that has few free parameters (simple controller) and uses a few numbers to sense the state and command the actuation (low bandwidth). Such an approach was used for the 65 km walk of the Cornell Robot Ranger [1,2]. This paper describes ideas used in [1] in a more basic way, and in the context of more simple examples.

Here we advertise classical, and well-known to some, discrete control as a (possibly under-appreciated) control architecture that is computationally simple and can accommodate time delays. We do not think of discrete control as an approximation for smooth continuous control, with sensing and actuation loops as fast as the computer system can handle. Rather, we think of the control as essentially discrete, with the spacing of control events possibly being much longer than the computer-bandwidth-limited sensing and acting loops. The times between control decisions may be comparable to, or even longer than, characteristic instability

times in the underlying system being controlled (the plant). The general approach here has been both thoroughly described, analyzed, generalized and advertised by Gawthrop and others [3–9]. Here, we present a simple description of a subset of the ideas in Gawthrop (and colleagues), and two simple examples.

Here is the basic Discete Decision Continuous Actuation (DDCA) approach.

I. Control occurs in intervals.
II. Each interval starts with an 'event'. An event could be triggered by the
   (a.) passage of time,
   (b.) a discrete event in the system (e.g., a collision),
   (c.) a threshold in some state being reached,
   (d.) a computational delay, or
   (e.) a mixture of these.
III At the start t_n of the interval n the state estimate x_n is read. This state estimate need not be up to date, but may lag by any known amount.
IV The discrete decision n is made. This is the evaluation of the control parameters U_n. In the *linear* cases we consider here, these are amplitudes to be multiplied by shape functions so that the control outputs are

$$u_n(t-t_n) = \sum_i U_{ni} * f_i(t-t_n).$$

The policy is then expressed by the gain matrix $\mathbf{K}$, which is used to calculate

$$U_n = -\mathbf{K}x_n.$$

For the *non-linear* cases,

$$u_n(t-t_n) = f(U_n, t-t_n)$$

and the policy is the function f.

From the instant of allowed actuation (starting after any sensing, computation and actuation lag), until the start of the next actuation, the control actions are governed by the essentially feedforward command $u_n(t-t_n)$. The "feedforward" commands could themselves have feedback. We just consider any real-time (or much-shorter than characteristic system time) state feedback within a control interval to be part of the definition of the plant and actuation.

This architecture is a special case of some control approaches and a generalization of some others. Even if the controller is designed using offline optimization, and thus could be called optimal-policy control, it uses only sparse and approximate evaluation of the true optimal control policy. The approach is like model-predictive control, but without updates, and with the model-optimization being pre-calculated off-line. And the differential equations of a model-based estimator are replaced with a simple map (a matrix multiplication in the linear case). It is discrete control, specialized to the case of continuous systems with continuous activation.

In the examples here we use a linear policy. That is, relative to sensing and goal states on a nominal uncontrolled trajectory, we assume a linear map from state deviations and control actions to the goal states. The linear map is most-conveniently calculated using a sensitivity computation (finite difference based differentiation) which we explain in Sec. 2.

As noted, our control idea seems nearly identical to the intermittent predictive control idea presented by Gawthrop *et al.* with some minor specializations and differences:

1. All computations, but for a single matrix multiplication once per control interval, are done offline;
2. The events are not necessarily varied with on-the-fly calculations; but are, in the examples here, pre-defined (with either time or state triggers); and
3. The net control architecture is simple enough so that it can be manually tuned. For example, what and when to sense, when to control, what 'hold' functions [10] (we call them shape functions in this paper) to choose. For example, our shape functions need not be based on LQ optimization.

The approach shares these differences with the dead-beat control approach promoted in [11], of which it is a slight generalization in that we consider non-constant, possibly overlapping in time, shape functions and more general approaches to finding gain matrices (e.g., LQR or intuitive, not just dead-beat).

Also, the method is somewhat similar in spirit to Insperger and Gabor's act and wait control in which the controller has a wait time (the control signal is zero) followed by an act time (the control signal is finite) [12, 13].

Open-loop discrete control The 'posicast' control method developed by Smith [14, 15] is an early implementation of discrete dead-beat control. In the posicast method, part of the input command is delayed to achieve dead-beat control. The delay is so chosen that the delayed input signal cancels the vibration produced by the pre-delay input signal.

The 'input shaping' or command shaping [16, 17] is yet another method to do dead-beat control. In input shaping, two impulses separated in time are convolved with the input command. The timing of the impulse is so chosen that the vibration induced by the first one is exactly cancelled by the second one leading to a dead-beat control.

Both methods above, the posicast and the input shaping method, tune the feed-forward command to achieve dead-beat control. At the discrete-control level this feedforward is really feedback, as the commands are based on the state at the start of the interval. The method here is more general and naive. The control here has the same general structure, but we do not promote any particular algorithms for picking optimized shape functions.

Controlling system with substantial delays. As noted, for continuous systems with delayed sensing or actuation,

it is commonly said that real-time (with delay) state based feedback can't stabilize a system whose natural instabilities have a characteristic time shorter than the delay time [13, 18, 19]. This is especially relevant in, say, models of animal control which need to account for the slowness of chemically-based nerve conduction, of neural computation, and of delays in muscle activation. In robotics delay can be substantial if either state-estimation or control calculations are large and the system has limited computational speed. One feature of the discrete control architecture is that it is not subject to such limitations; given sufficient sensor and actuation accuracy, the controller can stabilize an unstable system, even with long sensor delays. This ability to contend with time delays, discrete control shares with controllers using model-based state estimators [14, 20, 21] and also with Insperger's act and wait control [12, 13].

2 A simple discrete-control approach

2.1 Control problem

Let the state of the full, possibly non-linear, system be $x(t)$, the control be $u(t)$ and the continuous system dynamics defined by F with $\dot{x} = F(x,u)$. Further, assume the system has a desirable nominal trajectory $\bar{x}(t)$ associated with a nominal baseline control $\bar{u}(t)$:

$$\dot{\bar{x}} = F(\bar{x},\bar{u}). \tag{1}$$

The feedforward command $\bar{u}(t)$ in the above equation is open loop and does not stabilize the system adequately, or perhaps at all. For example, even with perfect initial conditions, modeling errors, actuator imperfections and disturbances will cause the system to too-much, or catastrophically ('failure'), deviate from the nominal trajectory.

Lacking a way to design a desirable stable open-loop controller, one needs feedback. The feedback controller supplements u with a control δu that adequately brings the system back to the nominal trajectory. In this case, we do feedback at discrete times and the control commands are simple feedforward control functions over the interval. This differs from common continuous feedback control because we only sense key quantities and only at occasional times.

2.2 Schematic example

We illustrate the event-based intermittent feedback control idea with a schematic example. Consider the nominal trajectory of a second-order system shown as a solid red color line in Fig. 1. Let n and $n+1$ be instances of time at which we are taking measurements from sensors. The time interval between the measurements n and $n+1$ is typically on the order of the characteristic time scale of interest (and *not* the shortest time our computational speed allows). Let us assume that we take two measurements, $x_n = [x_1 \;\; x_2]'$ (e.g., a position and velocity) at time n. We want to regulate two outputs: z_1 and z_2 (some attributes of the state x_n) at time $n+1$.

Assume that, due to external disturbances, the system has deviated from its nominal trajectory. We show the trajectory as a dashed blue color line in Fig. 1 (a). Now, the state of the system is $\bar{x}_n$ ($\neq x_n$) at time n. When feedback corrections are absent, the relevant output $\bar{z}_{n+1}$ ($\neq z_{n+1}$) whose components, in notational shorthand, are $[\bar{z}_1 \quad \bar{z}_2]'$.

Our feedback controller measures deviations at time n ($\delta x_n = x_n - \bar{x}_n$) and uses actuation to reduce the deviations in output variables ($\delta z_{n+1} = z_{n+1} - \bar{z}_{n+1}$). For illustration, we choose two control actions, $\delta u_n = [U_1 f_1(t) \;\; U_2 f_2(t)]'$, a half sinusoid and a hat function, each active for half the time between time $n+1$ and n (Fig. 1 (c)). The controller adjusts the amplitudes (U_1 and U_2) of the two control functions, based on measured deviations δx_n, to regulate the deviated outputs δz_{n+1}. For example, with a proper choice of the amplitudes, it should be possible to fully correct the deviations in the output variables, as seen in Fig. 1 (b).

In the simplest cases, we linearize the map from the measurement section n to the section $n+1$. The sensitivities of the dynamic state to the previous state and the controls $U_n = [U_1 \;\; U_2]'$ are: $\mathbf{A} = \partial x_{n+1}/\partial x_n$, $\mathbf{B} = \partial x_{n+1}/\partial U_n$, $\mathbf{C} = \partial z_{n+1}/\partial x_n$ and $\mathbf{D} = \partial z_{n+1}/\partial U_n$. The brute-force way of calculating the sensitivity matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\mathbf{D}$ is by numerical finite-difference calculations. We then have, for our linearized discrete system model:

$$\delta x_{n+1} = \mathbf{A}\delta x_n + \mathbf{B}U_n \tag{2}$$

$$\delta z_{n+1} = \mathbf{C}\delta x_n + \mathbf{D}U_n. \tag{3}$$

Again, the δx_n are a list of measured deviations, the δz_n are a list of deviations which we wish to control, the U are the activation amplitudes (2 in our example above). For simplicity, assume full state measurement, the controller architecture is thus

$$U_n = -\mathbf{K}\delta x_n, \tag{4}$$

where $\mathbf{K}$ is a constant gain matrix. We can choose the gains $\mathbf{K}$ to meet or optimize various goals using, say, pole placement or discrete linear quadratic regulator (DLQR) design.

Pole placement This method of control applies only to cases in which the goal variables z are the same as the state variables x. The controlled system dynamics are now

$$\delta z_{n+1} = \delta x_{n+1} = (\mathbf{A} - \mathbf{B}\mathbf{K})\delta x_n \tag{5}$$

The goal of pole placement is to choose the gain $\mathbf{K}$ so that eigenvalues of the closed system $(\mathbf{A} - \mathbf{B}\mathbf{K})$ have the desired values. For example, to place the eigenvalues at the origin, thus making a one-step dead-beat controller, and assuming the necessary invertibility of $\mathbf{B}$, we would have

$$\mathbf{K} = \mathbf{B}^{-1}\mathbf{A}. \tag{6}$$

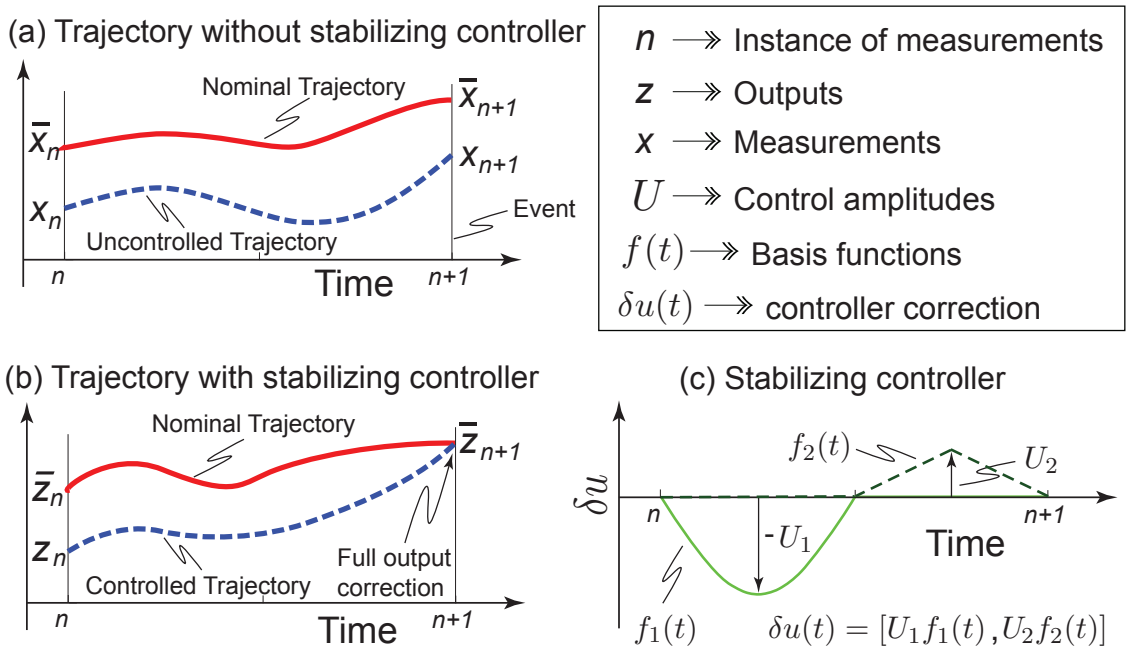

Fig. 1. **Schematic example. (a)** Shows the nominal (solid red) and deviated (dashed blue) trajectory, for some dynamic variable x of interest. We measure the state x at the start of a continuous interval, namely at section n. **(b)** Shows the new deviated trajectory in target variables z after switching on our feedback controller. In this example, feedback controller nulls (zeros) the output z at the end of the interval, illustrating a 'dead-beat' controller. **(c)** The feedback motor program has two control actions: a sinusoid for first half cycle and a hat function for the second half of the cycle. These shapes are arbitrary and different from each other in form only for illustrative purposes. They could overlap in time. We choose the amplitudes U_1 and U_2 of the two functions at the start of the interval depending on the error $(x-\bar{x})$. By a proper choice of the amplitudes U_1 and U_2 deviations are, in this example, fully corrected in between measurements. The choice of trigger for event n, the choice of sensor measurements x, the choice of output variables z, and the control shape functions $f(t)$ are offline design choices.

For most systems, ones that have the needed controllability, it is possible to find shape functions $f_1(t)$ and $f_2(t)$ so that the matrix $\mathbf{B}$ is non-singular. In the same way that a square matrix is generically non-singular, n random shape functions for an n order system should (generically) lead to a non-singular $\mathbf{B}$ and thus the possibility of 1-step dead-beat control. Of course the matrix $\mathbf{B}$ can be more or less well conditioned depending on how independent the shape functions are from each other.

Discrete linear quadratic regulator (DLQR) Another method uses DLQR and applies to any goal function z of the state. In DLQR [22], we seek to minimize the cost function J_{dlqr} defined as,

$$J_{\text{dlqr}} = \sum_{n=0}^{n=\infty} \left(\delta z_{n+1}{}^T \mathbf{Q}_{zz} \delta z_{n+1} + U_n{}^T \mathbf{R}_{UU} U_n \right), \qquad (7)$$

where $\mathbf{Q}_{zz}$ and $\mathbf{R}_{UU}$ are matrices that weight the different components of δz_{n+1} and U_n ($\mathbf{R}_{UU}$ must be positive definite and $\mathbf{Q}_{zz}$ positive semi-definite). The weights $\mathbf{Q}_{zz}$ and $\mathbf{R}_{UU}$ are design parameters picked to give reasonably fast return to nominal values but without unduly high gains (which might tend to lead to control command that are beyond safety limits). They are often given as diagonal for simplicity.

Putting Eqn. (3) in Eqn. (7) and re-arranging gives,

$$J_{\text{dlqr}} = \sum_{n=0}^{n=\infty} \left(\delta x_n{}^T \mathbf{Q} \delta x_n + 2 \delta x_n{}^T \mathbf{N} U_n + U_n{}^T \mathbf{R} U_n \right), \qquad (8)$$

where $\mathbf{Q} = \mathbf{C}^T \mathbf{Q}_{zz} \mathbf{C}$, $\mathbf{N} = \mathbf{D}^T \mathbf{Q}_{zz} \mathbf{C}$ and $\mathbf{R} = \mathbf{D}^T \mathbf{R}_{zz} \mathbf{D} + \mathbf{R}_{UU}$. J_{dlqr} can be minimized with a linear state feedback, $U_n = -\mathbf{K} \delta x_n$ with gain $\mathbf{K}$ found by solving the standard Ricatti equation [22]. We do this using MATLAB control system toolbox (DLQR).

Other goals. The same linear control architecture given by Eqn. 4, could have gains $\mathbf{K}$ chosen to optimize or achieve other criteria that do not fit into standard basic linear control formalisms. For example, there could be a weight on the sparseness of $\mathbf{K}$, on non-quadratic costs for error and control over some range of initial conditions, on the basin of attraction for the non-linear system, etc. To calculate $\mathbf{K}$ one might then require more involved optimization calculations, but the structure of the resultant controller would be preserved. Similarly the choice of shape functions could be subject to optimization on independence, smoothness, maximizing control

authority, etc.

Factors to consider while designing the controller: The systems we are interested in controlling are not those in which we do measure control quality by how closely a target is followed, clearly the type of intermittent control we discuss here is not optimal for that. Rather, we are interested in preventing total system failure. For walking or for an inverted pendulum, falling down is failure. To slightly generalize, by failure we mean that the system state has moved outside a particular target region surrounding the target point. How is this region defined? In practice, it is the region outside of which non-linear effects lead to divergence of the solution to points much farther from the target (e.g., falling down). Sticking to the linear model, the user has to supply the target region based on intuitions, experience, or non-linear modeling (all of which are outside the coverage of this paper).

Some issues in the controller design include:

1. Selecting a suitable section or instance of time to take measurements — this instant should be when the dynamic-state estimation is reasonably accurate, and when dynamic-state errors which cause failure are evident;
2. selecting measurement variables (x_n) that are well-predict system failure ;
3. picking output variables (z_n) that can well-correct against system failure; and
4. picking actuator shape profiles ($f(t)$'s) that have large, and relatively independent, effects on the target variables, and are also sufficiently smooth for implementation with real motors.

We next discuss the above points with in the context of a walking robot.

Example: Controlling a bipedal walking robot For a 2D bipedal robot walking at steady speed, here is how we can go about designing a discrete controller [23]. A typical walking step of a bipedal robots includes two phases: a smooth continuous phase in which the entire robot vaults over the grounded leg, and a non-smooth discontinuous phase in which the legs exchange roles.

1. Suitable section or instance of time to take measurements: Any instant not-close to support-exchange is a good time for measurement. This is because the measurements are typically noisy during the non-smooth support change (heel-strike collision).
2. Suitable measurement variables (x_n) that are representative of system failure: The state of the lower body is most important for walking balance, so good measurement variables are the state (position and velocity) of the stance leg.
3. Suitable output variables (z_n) that also correlate with system failure: Step time, step length are important quantities to regulate during walking, and they serve as good output variables.
4. Suitable actuator shape profiles ($f(t)$'s) that have large and relatively independent effects on the target variables: For leg swing, for example, two torque profiles, one with large amplitude near the start of the interval, and one with large amplitude near the end, yield good control authority over position and velocity of the swing leg at the end of the interval.

Once the above quantities are picked, we can check the system controllability. If the system is not well controllable (correction of reasonable disturbances requires unreasonable actuation amplitudes) the first likely fix is picking better actuation shape functions.

As noted, we used this discrete feedback control idea to stabilize steady walking gait of a bipedal robot leading to energy-efficiency record and long distance 65 km walking record [1, 2, 24].

2.3 Computing the linearization

For linear control approaches, the gain selection depends on having the linearized map Eqn. (2) and Eqn. (3) from Eqn. (1). We assume we have a system, or computational model of the system, with which we can perform numerical experiments. To get the matrices **A** and **C**, we can perturb x_n element-wise and use finite difference to compute these matrices. Similarly to get matrices **B** and **D**, we can put in small amplitudes of the controls U_n and use finite difference to compute the sensitivities.

3 Balance of a simple inverted pendulum

Amongst the simplest mechanical control problems is balance of a simple inverted pendulum with a stationary hinge at its base. Our experimental system has a pendulum length of length, $\ell \sim 1$m, and hence a characteristic instability time of $\sqrt{\ell/g} \approx 0.32$s. To highlight a benefit of discrete control, we will sense once per second, three times longer than the characteristic instability time.

3.1 Inverted pendulum model

For simplicity we start with the linearized equations. The inverted pendulum shown in Fig. 2 (a) has this linearized governing equation:

$$\dot{x} = F(x,u) = \mathbf{a}x + \mathbf{b}u, \tag{9}$$

where $\theta = x_1$, $\dot{\theta} = x_2$, $u = T_m$, $\mathbf{a} = \begin{bmatrix} 0 & 1 \\ \frac{g}{\ell} & 0 \end{bmatrix}$, and $\mathbf{b} = \begin{bmatrix} 0 \\ \frac{1}{m\ell^2} \end{bmatrix}$,

3.2 Balance Controller

Next, we design an event-based feedback controller (see Sec. 2). As noted, successful stabilizing control depends on good-enough selection of four key quantities: a suitable section or instance of time to take measurements, measurement variables (x_n), output variables (z_n) and nature of controls actions (U_n). We discuss these next.

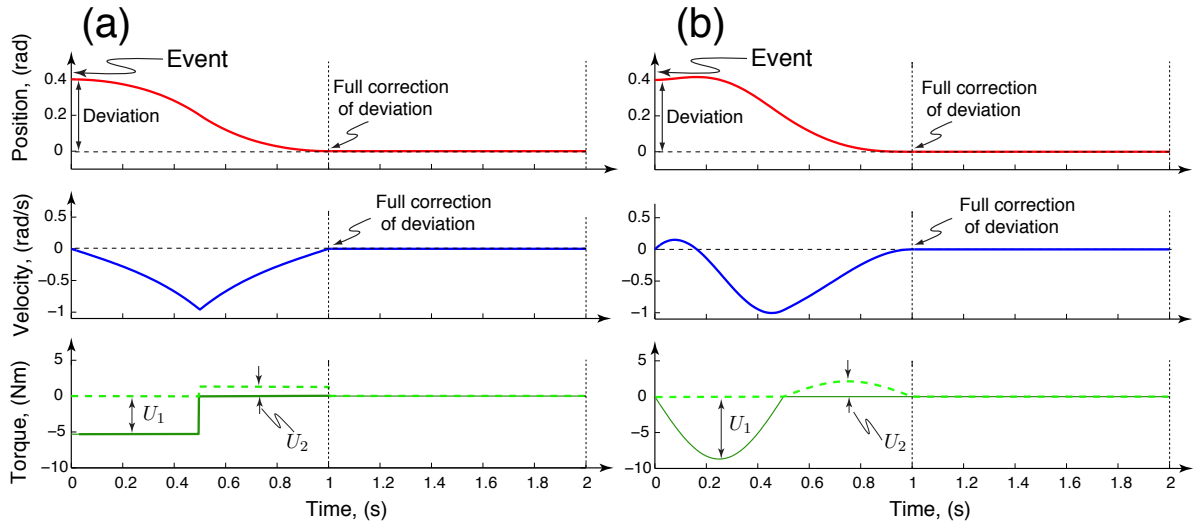

Fig. 3. **Simulation results for balance of a simple inverted pendulum** The first column (a) uses constant control function and second column (b) uses sinusoidal control functions. The first, second and third row correspond to joint angles vs time, joint rate vs time and control torque vs time, respectively. In both cases the system is let go from an initial position of 0.4 rad and initial velocity of 0. We used the linearized governing equation to do these simulations.

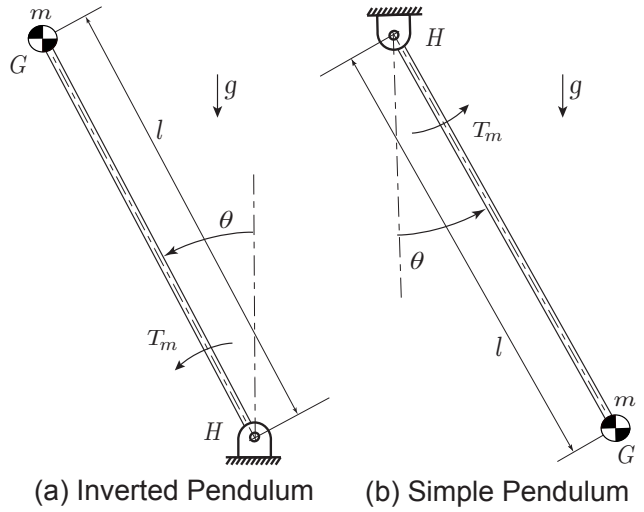

Fig. 2. **(a) Inverted pendulum and (b) Simple pendulum.** The pendulum in both cases consists of mass m at G and attached to a massless rod of length ℓ and controlled by a motor via a torque T_m at the hinge joint H. Gravity is g.

Here we base our 'event' on the passage of time, we take measurements once every $T = 1$s. We choose our output variables z_n to be the same as the measurement variables x_n, i.e. the joint angle and the joint velocity. We choose two control actions, $\delta u_n = [U_1 f_1(t) \;\; U_2 f_2(t)]'$, each active for half the time between time $n+1$ and n. Later we will make specific choices for the control functions $f_1(t)$ and $f_2(t)$. Also note that difference between scalar u in Eqn. (9) and vector δu_n. The former is the instantaneous motor torque while the latter is a set of discrete control actions (also in the units of motor torques) that are functions of time. Each action is active for $T = 1$s each.

Our nominal trajectory has $x = 0, u = 0$. Putting $\mathbf{A} = \partial x_{n+1}/\partial x_n$, $\mathbf{B} = \partial x_{n+1}/\partial U_n$, where $U_n = [U_1 \;\; U_2]'$ and noting that $x_{n+1} = z_{n+1}$ we get,

$$\delta z_{n+1} = \delta x_{n+1} = \mathbf{A}\delta x_n + \mathbf{B}U_n. \tag{10}$$

Note that the instantaneous motor torque, u, is implicitly present through U_n when we did the substitution for δu_n.

For this linearized example, we find the matrices $\mathbf{A}$ and $\mathbf{B}$ analytically,

$$\mathbf{A} = e^{\mathbf{a}T} \tag{11}$$

$$\mathbf{B} = \left[\int_{\tau=0}^{\tau=\frac{T}{2}} e^{\mathbf{a}(T-\tau)}\mathbf{b}f_1(\tau)d\tau, \int_{\tau=\frac{T}{2}}^{\tau=T} e^{\mathbf{a}(T-\tau)}\mathbf{b}f_2(\tau)d\tau\right]. \tag{12}$$

As an example, we will use pole placement and find a dead-beat controller of the form $U_n = -\mathbf{K}\delta x_n$, where $\mathbf{K}$ is a gain matrix and put this in Eqn. (10) to get,

$$\delta z_{n+1} = \delta x_{n+1} = (\mathbf{A} - \mathbf{BK})\delta x_n \tag{13}$$

In pole placement we find a gain $\mathbf{K}$ so that eigenvalues of the closed system $(\mathbf{A} - \mathbf{BK})$ are what we want, in this case all zeros. As noted above, the gain matrix $\mathbf{K}$ this dead-beat control is then given by, $\mathbf{K} = \mathbf{B}^{-1}\mathbf{A}$.

A comment on linearity. If the plant is linear then our resulting system is linear, assuming we use a time-based event. If the event trigger is state-based then the system is still linear, *starting at* the event and until the next event. However,

despite the linear calculations in the system and the controller, in the case of event-based control the overall system is *non-linear*. For example, doubling an initial disturbance would not then double the controlled system's response. See Sec. 4 for an example of a linear system that becomes nonlinear, after doing the linearization in state and control, because of a state-based (rather than time-based) event.

3.3 Results

We set up our experiment with a 1 m long rod with mass of 1 kg attached to its end. For control design we used $g = 10$ m/s^2, $\ell = 1$ m, $m = 1$ kg. We choose the sampling time $T = 1$ s for control. Note again that the characteristic time scale of this simple pendulum is about 0.32 s (time scale = $\sqrt{\ell/g} = \sqrt{1/10}s \approx 0.32s$), which about 3 times faster than the sampling time of 1 second that we will be using.

Eigenvalue of uncontrolled system. Putting

$$\mathbf{a} = \begin{bmatrix} 0 & 1 \\ 10 & 0 \end{bmatrix}, T = 1\text{s in Eqn. (11) gives, } \mathbf{A} = \begin{bmatrix} 11.83 & 3.72 \\ 37.28 & 11.83 \end{bmatrix}.$$

The biggest eigenvalues of the system is $\approx 23.6/\text{s}$. This implies that disturbance to the system at time $t = 0$, will grow by up to a factor of 23.6 in 1 second.

For illustration we use two different pairs of functions of time: **a)** $f_1(t) = 1$ for the first half of the interval and zero in the second half, and $f_2(t) = 1$ in the second half and zero in the first; and **b)** $f_{1,2}(t) = \sin(2\pi t/T)$ but with each function zero in differing halves of the control interval.

Example 1: Using constants for control functions. We put

$$\mathbf{a} = \begin{bmatrix} 0 & 1 \\ 10 & 0 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, T = 1, \text{and } f_1(t) = f_2(t) = 1$$

$$\text{in Eqn. (12), to get } \mathbf{B} = \begin{bmatrix} 0.93 & 0.15 \\ 2.99 & 0.73 \end{bmatrix}.$$

Using the matrix $\mathbf{A}$ calculated earlier and value of $\mathbf{B}$ in equation $\mathbf{K} = \mathbf{B}^{-1}\mathbf{A}$, we calculated the

$$\text{gain matrix as } \mathbf{K} = \begin{bmatrix} 13.26 & 4.12 \\ -3.26 & -0.67 \end{bmatrix}.$$

Example 2: Using sinusoids for control functions. We put

$$\mathbf{a} = \begin{bmatrix} 0 & 1 \\ 10 & 0 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, T = 1, \text{and} f_1(t) = f_2(t) = \sin\left(\frac{2\pi\tau}{T}\right)$$

$$\text{in Eqn. (12), to get } \mathbf{B} = \begin{bmatrix} 0.56 & -0.09 \\ 1.82 & -0.4487 \end{bmatrix}.$$

Using the matrix $\mathbf{A}$ calculated earlier and value of $\mathbf{B}$ in equation $\mathbf{K} = \mathbf{B}^{-1}\mathbf{A}$, we calculated the

$$\text{gain matrix as } \mathbf{K} = \begin{bmatrix} 21.75 & 6.76 \\ 5.34 & 1.11 \end{bmatrix}.$$

Fig. 3 shows the joint angle, joint velocity, and commanded torque vs time for the simulated inverted pendulum with constant controls and sinusoid functions. In either case we disturb the system by setting the initial condition $x(0) = [0.4 \;\; 0]'$. Both the control functions are able to cancel the effect of the disturbance in one sampling step i.e. 1 sec. In other words, the system achieves dead-beat control in 1 sampling time step. However, note that the choice of the control functions affects the maximum torques used. While the constant function uses a max torque of about 5 Nm, the sinusoid uses a max torque of about 9 Nm.

In both cases the sensing is of 2 numbers (not two functions) per interval, the gains are 4 numbers, and the controller calculates 2 amplitudes (not 2 functions, we assume that the control functions are stored at a lower level), and the online calculation is a single matrix multiplication. This linear controller has achieved dead-beat control and stabilization of a system with time delay (time between sendings) longer than the control loop time.

Experimental Verification Fig. 4 shows the experiment. There is a mass of about 1 kg at the end of the 1 m long rod. We neglect the mass of the light carbon fiber rod in the controller design. We measure the joint angles by an incremental encoder. We calculate the joint angular rate from the joint angles using numerical differentiation followed by a low pass filter. The joint angle sensor had about 1° hysteresis error (dead band). The motors have a load-dependent friction and constant Coulomb friction. The gear backlash is about 2°. Using the controller gain, $\mathbf{K}$, obtained from the constant control function calculated above – which did not account for backlash, friction, or angle hysteresis – we were able to balance the inverted pendulum from initial perturbations up to $\pm$ 0.5 rad. Because of all the modeling errors and disturbances, the 'stabilized' motion was erratic with an amplitude of $\pm$ 0.15 rad. We expect this because in the $T = 1$s between sensor inputs, disturbances grow by about a factor of 23. A video of the experiment is available online [25].

4 Pumping a swing

We next present a controller that pumps the pendulum swing to make it oscillate at a given amplitude, say 1 rad. This example was also used in [11]. Here we use state-based triggering. We sense the velocity when the pendulum is vertically down and moving from left to right (see Fig. 2 (b)). We will pump the swing with an amplitude based on this sensed velocity.

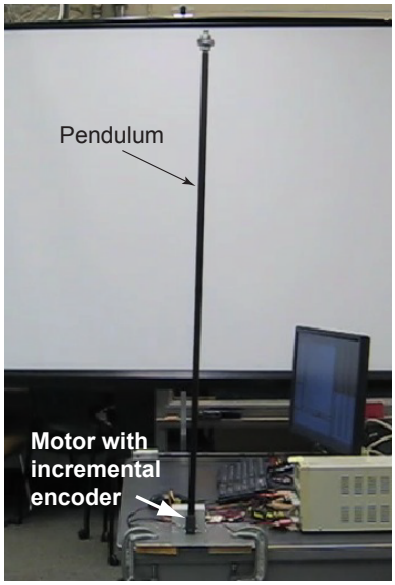

Fig. 4. **Experimental verification of event-based intermittent controller to balance a simple inverted pendulum.** We measured the pendulum state – the angle and angular speed – once per second and used constant control functions active for half a second each. We were able to balance the inverted pendulum over a range of ± 0.5 rad. Note that the sensing is 1 s, and the controller bandwidth is 0.5 s, and is slower than the characteristic time scale of 0.32 s of the simple inverted pendulum. See reference [25] for a video.

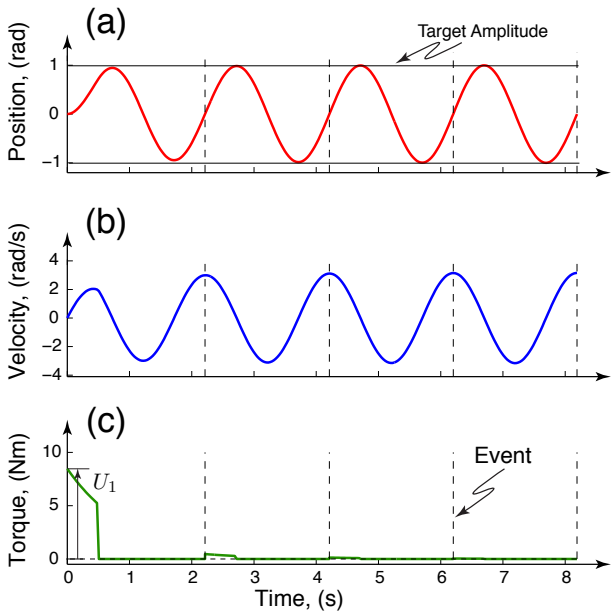

Fig. 5. **Simulated result for pumping the swing of a simple pendulum.** The pendulum starts from rest from the hanging position. (a) Pendulum Angle vs Time, (b) Pendulum Velocity vs Time, and (c) Torque vs Time.

4.1 Simple pendulum model

The equation of the simple pendulum shown in Fig. 2 (b) is given by

$$\dot{x} = F(x,u) = \mathbf{a}x + \mathbf{b}u \tag{14}$$

where $\theta = x_1$, $\dot{\theta} = x_2$, $u = T_m$, $\mathbf{a} = \begin{bmatrix} 0 & 1 \\ -\frac{g}{\ell} & 0 \end{bmatrix}$, and $\mathbf{b} = \begin{bmatrix} 0 \\ \frac{1}{m\ell^2} \end{bmatrix}$,

4.2 Swing controller

We take measurements every time the pendulum is in the vertical down $x_1 = 0$ and swinging counter-clockwise, that is, $x_2 > 0$. Our output z_{n+1} was the target amplitude of the pendulum when it is on the extreme right position. We choose one control action, $\delta u_n = U_1 f_1(t)$, where $f_1(t) = e^{-t}$ (an arbitrary choice) that we zero after 0.4s. Why .4s? Because that is long enough so that the actuator can have substantial effect even with low amplitude, but ends safely before a control reversal (the peak in the motion).

4.3 Simulation results

We consider a pendulum length of 1 m with mass of 1 kg attached to its end. We put $g = 10$ m/s^2, $\ell = 1$ m, $m = 1$ kg.

To compute the linearization in Eqn. (2) and Eqn. (3) we use a central difference (see Sec. 2.3) with a perturbation size of 0.001 to generate the A, B, C and D.

The Jacobian of the uncontrolled system, $A = 1.0$. This is a system with a neutral equilibrium i.e. the disturbances will neither grow nor diminish. We also get $B = 0.26$, $C = 0.31$, and $D = 0.08$. We used DLQR with $R_{UU} = 1$ and $Q_{zz} = 250$, to get $K = 2.68$. This gain corresponds to an eigenvalue of 0.28 for the controlled system.

Fig. 5 shows the effect of our linear controller on this non-linear system. We start the system from rest and with zero amplitude. The amplitudes for the subsequent four oscillations are 0.94, 0.98, 0.98 and 0.998. In other words, using our control, we are within 0.2% of desired amplitude in four oscillations.

5 Discussion

Time delays and linear continuous controllers. It is well known that linear continuous controllers (e.g. Proportional Derivative (PD) Controller that does feedback on position and velocity) have difficulty stabilizing system with substantial delays. In fact, the inverted pendulum presented in this paper cannot be controlled by a PD controller if the feedback delay is longer than $\sqrt{2\ell/g} \sim 0.45$ s [19]. Insperger et al. [26] claim that it is possible to increase the feedback delay by 40% (that is $\sim 1.4 \times 0.45 = 0.63$ s) by doing a feedback on position, velocity and acceleration.

Note that the characteristic equation of a time delayed system in continuous time has infinite poles [12]. However, by casting the problem in discrete time, we have finite number of poles which are often manageable. For example, using discrete control there is no fundamental limit on the magni-

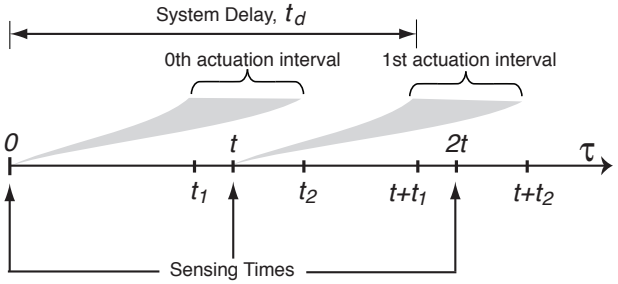

Fig. 6. **Definition of delay for discrete control of a continuous system.** One sensible definition of delay would be the time t_1 between when the system is sensed and when the start of the related control action. This does not respect an invariance described in the text. A sensible definition of delay is the time $t+t_1$ between the start of an actuation interval and the most recent sensing that had an effect, through previous actuation, on the starting state.

tude of the time delays in relation to the system instability times.

Definition of delay for discrete control. For a controlled continuous-system, time delay is the span between the instant of sensing and the start of any resulting control action. For a discretely controlled continuous system there is no simple definition of delay that combines delays due to sensing at intermittent times and delays due to transmission and computation times. For the purposes of discussion it is nice to have such a definition. For simplicity, assume a linear time-invariant system.

Let the system be sampled at time $0, t, 2t, \ldots$ (See Fig. 6). Due to essential transmission delays the associated control actions can start at time instances $t_1, t+t_1, 2t+t_1 \ldots$ Assume the actions must end at time instance $t_2, t+t_2, 2+t_2 \ldots$ Assume state feedback with action at any time depending only on the most recently sensed state. For example, in the interval $t_1 \leq \tau \leq t_2$ the control actions only depend on the sensing at time 0. Assume the interval of possible action for sensing at $\tau = 0$ is $0 \leq t_1 \leq \tau \leq t_2$ (In our examples, the actuation intervals match the sensing intervals in duration, so $t_2 = t+t_1$). We seek a definition of delay t_d in terms of t, t_1 and t_2.

Let us assume that, based on sensing at $t = 0$, we send an instantaneous actuation U_1 (e.g., impulsive) at time $\tau = t^-$, just before the next sample. For discrete control purposes we can write the control equation between the two time sampling intervals as

$$\delta x_{n+1} = A\delta x_n + B_1 U_1 \quad \text{Action } U_1 \text{ at end of sample interval} \tag{15}$$

One could say this system has delay t because that is the time between sensing and action.

Now consider the same system, but with the impulsive actuation U_2 at the start of the sampling interval, i.e. at $\tau = 0^+$. That system has no delay between sensing and acting. In this case, the discrete control equation between the two sampling intervals is

$$\begin{aligned} \delta x_{n+1} &= A\left[\delta x_n + B_1 U_2\right] \quad \text{(Action } U_2 \text{ at start of sample interval).} \\ &= A\delta x_n + AB_1 U_2 \\ &= A\delta x_n + B_2 U_2 \end{aligned} \tag{16}$$

We assume B_1 is invertible (which it must be for, say, single-step dead-beat control). Then if the controls U_1 and U_2 are related by $U_1 = B_1^{-1} A B_1 U_2$ the two systems above have identical dynamics. Also equivalent to these is any system with continuous actuation occurring at any time in the interval 0 to t. Thus these two systems should sensibly be described as having the same delay.

Alternate definition of delay. Two restrictions on our definition of delay are that

1. For the continuous limit it has to agree with notions of delay for continuous systems. So for the continuous special case, $t = 0$ and $t_1 = t_2$, the formula for delay must give $t_d = t_1 = t_2$.
2. The definition needs to give the same answer independent of when in the available interval $t_1 \leq \tau \leq t_2$ the controller chooses to use the actuation (see examples above).

These two candidate definitions pass these reasonability tests:

1. t_d is the time from sensing to first action. However, this doesn't give the same delay for the two equivalent systems defined above.
2. t_d is the average of the action interval. However, this doesn't have the property of equivalence between the choice of acting at the start of the interval or at the end.

A definition that meets the basic constraints is that the effective feedback delay is from sensing until the start of the next actuation interval. That is, the effective definition of delay we favor is

$$t_d = t + t_1.$$

For our first examples, where sensing and computation are instantaneous, but action is in the intervals between sensing, the effective delay is the sampling interval, $t_d = t$. The definition above for delay answers the question, "Just before the start of the present period of activation, what was the most recent sensing that had effect, through previous activation, on the present state?"

Exponential sensitivities caused by time delay If the uncontrolled system is unstable, unlike for continuous real-time control, time delays do not cause instability. Rather, however, imperfections in the model or sensing cause errors which grow exponentially with the amount of delay with only a critical-sized sensor error being tolerable. Consider

the observable controlled system with a dead-beat controller over intervals with span t:

$$\dot{x} = ax + u \qquad \text{continuous system} \quad (17)$$

$$\delta x_{n+1} = (\mathbf{A} - \mathbf{BK})\delta x_n \qquad \text{discrete system} \quad (18)$$

$$\mathbf{A} = e^{at} = \mathbf{BK} \qquad \text{linear dead-beat controller.} \quad (19)$$

Now consider a sensing error δx_s so that

$$\delta x_{n+1} = \mathbf{A}\delta x_n - \mathbf{BK}\delta x_n + \mathbf{A}\delta x_s \quad (20)$$

$$= \mathbf{A}\delta x_s = e^{at} x_s. \quad (21)$$

Thus, even with perfect dead-beat control design noise is amplified. Say the largest eigenvalue of a is λ = (system instability time)$^{-1} \equiv t_i$ thus

$$\text{Response error} = (\text{System error}) * e^{\frac{t}{t_i}}. \quad (22)$$

Similar reasoning applies to modeling errors Δa, so that the actual system and control are:

$$\delta x_{n+1} = (e^{(a+\Delta a)t} - \mathbf{BK})\delta x_n \qquad \text{discrete system} \quad (23)$$

$$\mathbf{BK} = e^{at} \qquad \text{controller not based on model errors.} \quad (24)$$

For order-of-magnitude calculation, replacing a with it's largest eigenvalue and assuming we can do the same with Δa, and approximating $e^{\Delta at} \approx 1 + \Delta at$ we get

$$x_{n+1} = \Delta a t e^{at} x_n = \frac{\Delta a}{a} a t e^{at} x_n = \frac{\Delta a}{a} (t/t_i) e^{t/t_i} x_n. \quad (25)$$

Equations (22)-(25) are the essential, unavoidable, control problem for systems with delay. It is not that such systems cannot be stabilized. It is that the magnitude of the errors, after implementing the best possible controls, grow exponentially with the length of the delays. For example, given a discrete time three times the instability time ($t = 3t_i$) a 10% error in sensing ($\delta x_s = 0.1 x_s$) leads to an output noise of $0.1 * e^3 \approx 2.3$ times the original signal error. Thus sensing errors (assumed proportional to signal size) with

$$\delta x_s > \exp^{-t/t_i} \delta x$$

can lead to a sensor caused instability. Similarly for modeling errors. The largest tolerable modeling error is

$$\frac{\Delta a}{a} = \frac{e^{-t/t_i}}{t/t_i}.$$

For example, if $t/t_i = 3$ then the largest tolerable modeling error is $\approx e^{-3}/3 \approx 1/69$. Thus control of unstable systems with significant time delay cannot be robust to sensing noise or modeling errors, by the means presented here, or by any other means,

In systems with intrinsic time delays but faster computations, the approach here may have little penalty over real-time model-based methods. However, in systems without intrinsic substantial time delays, the method here artificially introduces time delays, and thus our controls unavoidably increase, relative to a good continuous control, sensitivity to modeling errors and disturbances. These are the penalties for the simple design and reduced bandwidth.

Discrete control to achieve dead-beat control. Dead-beat control refers to full correction of deviation in finite time (e.g., see [27] page 201). The ability to achieve dead-beat linear control is unique to discrete time control; continuous linear control (e.g., proportional or proportional-derivative control) always leads to exponentially decaying response (e.g., see [22] page 416-417).

Minimal online computation. Unlike realtime model-based systems this approach needs neither real-time solution of ordinary differential equations nor online optimization. The online calculations needed are: 1) Continuous checking of the state-based or clock driven event; 2) Once-per-discrete-interval calculation of the control parameters, in the linear case this is a single matrix multiplication; 3) Real time calculation of the control shape functions (in the linear case this is a control gain multiplied by a given shape function). That is, the approach here uses far less computation than any real-time optimization or model-based control approach (which use an optimization calculation at each event). The approach also somewhat simplifies the model-based state estimation problem in that the state estimate is allowed to lag in time,

Under-actuated vs fully actuated. As is a core topic in control theory, a system that has fewer independent actuators than degrees of freedom can often be controllable. In the discrete domain, such a system might in some sense become fully actuated. That is, the system can achieve any state, starting from any other, in one time interval if the n shape functions are independent. Thus, walking with point feet, a famously under-actuated system, is fully actuated in that all degrees of freedom at the Poincare sections can often be independently controlled in one physical step.

6 Conclusions

In this paper, we developed a simple control framework for continuous systems that only have occasional, or perhaps loose, control goals. The controller collects sensor data at prescribed points in state space or time (event-based feedback) and uses these measurements to develop control commands that are feedforward for prescribed time (intermittent control) to regulate key quantities in a trajectory. This Discrete Decision Continuous Actuation intermittent control approach has the following advantages; 1) It is intuitive; 2) It

can seamlessly incorporate time delays; 3) It is computationally inexpensive as all gains and set points are computed offline; 4) It is simple, as the control design uses few numbers (gains and set points) to track a few key quantities; 5) It can do full correction of disturbances (dead-beat control) using a linear control law for a linear plant. The price paid for these gains is less-than-optimal tracking, even of intermittent goals.

This control approach highlights the essential problem of control with delays. Delays are well-known to be problematic. But it is not true that systems with long delay, say longer than plant instability times, cannot be stabilized. They can be. Methods to do this include model-based estimation, using act and wait, or using discrete control of the type described here. Rather, the essential problem with system delays is that the errors in the controlled plant state necessarily grow exponentially in the delay time. Thus, to practically control a system with long delays requires, relative to the goal accuracy, exponentially small model errors, exponentially small sensing errors and exponentially small system disturbances (exponential in the delay time).

On the other hand, if the model is good and expected disturbances are small, time delay is simply accommodated.

Acknowldegements

The authors would like to thank Jason Cortell, Nicolas Williamson, and Thomas Craig who developed the pendulum platform and to anonymous reviewers, whose comments led to substantial improvements in the text. This research was supported by an NSF grant number 52836 to Andy Ruina.

References

[1] Bhounsule, P. A., Cortell, J., Grewal, A., Hendriksen, B., Karssen, J. D., Paul, C., and Ruina, A., 2014. "Low-bandwidth reflex-based control for lower power walking: 65 km on a single battery charge". *The International Journal of Robotics Research,* **33**(10), pp. 1305–1321.

[2] Ruina, A., 2012. Cornell ranger 2011, 4-legged bipedal robot. `http://ruina.tam.cornell.edu/research/topics/locomotion\_and\_robotics/ranger/Ranger2011/` or google search for cornell ranger, April 2012.

[3] Gawthrop, P. J., and Wang, L., 2006. "Intermittent predictive control of an inverted pendulum". *Control Engineering Practice,* **14**(11), pp. 1347–1356.

[4] Gawthrop, P. J., and Wang, L., 2007. "Intermittent model predictive control". *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering,* **221**(7), pp. 1007–1018.

[5] Gawthrop, P. J., and Wang, L., 2009. "Event-driven intermittent control". *International Journal of Control,* **82**(12), pp. 2235–2248.

[6] Gawthrop, P., Wagg, D., Neild, S., and Wang, L., 2013. "Power-constrained intermittent control". *International Journal of Control,* **86**(3), pp. 396–409.

[7] Gawthrop, P., Lee, K.-Y., Halaki, M., and ODwyer, N., 2013. "Human stick balancing: an intermittent control explanation". *Biological cybernetics,* **107**(6), pp. 637–652.

[8] Gawthrop, P., Loram, I., Gollee, H., and Lakie, M., 2014. "Intermittent control models of human standing: similarities and differences". *Biological cybernetics,* **108**(2), pp. 159–168.

[9] Gawthrop, P., Gollee, H., and Loram, I., 2014. "Intermittent control in man and machine". *arXiv preprint arXiv:1407.3543*.

[10] Gawthrop, P., 2010. "Act-and-wait and intermittent control: some comments". *IEEE transactions on control systems technology,* **18**(5), pp. 1195–1198.

[11] van der Linde, R. Q., 1999. "Design, analysis, and control of a low power joint for walking robots, by phasic activation of mckibben muscles". *IEEE Transactions on Robotics and Automation,* **15**(4), pp. 599–604.

[12] Insperger, T., 2006. "Act-and-wait concept for continuous-time control systems with feedback delay". *Control Systems Technology, IEEE Transactions on,* **14**(5), pp. 974–977.

[13] Insperger, T., and Stépán, G., 2007. "Act-and-wait control concept for discrete-time systems with feedback delay". *Control Theory & Applications, IET,* **1**(3), pp. 553–557.

[14] Smith, O. J., 1959. "A controller to overcome dead time". *ISA Journal of Instrument Society of America,* **6**(2), pp. 28–33.

[15] Hung, J. Y., 2007. "Posicast control past and present". *IEEE Multidisciplinary engineering education magazine,* **2**(1), pp. 7–11.

[16] Singer, N. C., and Seering, W. P., 1990. "Preshaping command inputs to reduce system vibration". *Journal of Dynamic Systems, Measurement, and Control,* **112**(1), pp. 76–82.

[17] Singhose, W., 2009. "Command shaping for flexible systems: A review of the first 50 years". *International Journal of Precision Engineering and Manufacturing,* **10**(4), pp. 153–168.

[18] Bhatt, S., and Hsu, C., 1966. "Stability criteria for second-order dynamical systems with time lag". *Journal of Applied Mechanics,* **33**(1), pp. 113–118.

[19] Stépán, G., 1989. *Retarded dynamical systems: stability and characteristic functions*, Vol. 200. Longman Scientific & Technical Essex.

[20] Fuller, A., 1968. "Optimal nonlinear control of systems with pure delay?". *International Journal of Control,* **8**(2), pp. 145–168.

[21] Kleinman, D. L., 1969. "Optimal control of linear systems with time-delay and observation noise". *Automatic Control, IEEE Transactions on,* **14**(5), pp. 524–527.

[22] Ogata, K., 1995. *Discrete-time control systems*. Prentice-Hall Englewood Cliffs, NJ.

[23] Bhounsule, P. A., 2014. "Control of a compass gait walker based on energy regulation using ankle push-off and foot placement". *Robotica,* **FirstView**, 7, pp. 1–11.

[24] Bhounsule, P. A., 2012. "A controller design framework for bipedal robots: trajectory optimization and event-based stabilization". PhD thesis, Cornell University, Ithaca, NY, USA.

[25] Bhounsule, P., and Ruina, A., 2013. Feedback control of a time delayed inverted pendulum. `http://youtube.com/watch?v=GCGeDHKNzm4` or `http://tiny.cc/pranavb_delay`, November 2013.

[26] Insperger, T., Milton, J., and Stépán, G., 2013. "Acceleration feedback improves balancing against reflex delay". *Journal of The Royal Society Interface,* **10**(79), p. 20120763.

[27] Antsaklis, P. J., and Michel, A. N., 2006. *Linear systems*. Springer, Birkhauser, Boston, MA.