Control-On-Request: Short-Burst Assistive Control for Long Time Horizon Improvement

Alex Ansari and Todd Murphey

***Abstract*— This paper introduces a new control-on-request (COR) method that improves the capability of existing shared control interfaces. These COR enhanced interfaces allow users to request on-demand bursts of assistive computer control authority when manual / shared control tasks become too challenging. To enable the approach, we take advantage of the short duration of the desired control responses to derive an algebraic solution for the optimal switching control for differentiable nonlinear systems. These solutions minimize control authority while generating immediate and optimal degrees of improvement in system trajectories. The technique avoids the iterative constrained optimization required to derive traditional finite horizon optimal switching control solutions for nonlinear system and allows optimal control responses on time-frames otherwise infeasible. An interactive example illustrates how users can use COR interfaces to request real-time bursts of control assistance to help stabilize an unstable system.**

I. INTRODUCTION

This paper focuses on a specific form of control that will be referred to as *burst control*. These control laws are defined as a subclass of switching control laws. They consist of a continuous nominal control mode that switches to alternate, continuous control modes lasting for short but finite durations (see Fig. 1). As an example, consider a pinball table where, nominally, the ball moves without control. By pushing buttons on either side of the table, a user actuates toggles that apply short duration controls of varying amplitude when they contact the ball.[1] In this case, users generate burst control signals to aim and influence the ball's trajectory.

A number of important systems directly apply burst control. For example, spacecraft and satellites use what can be modeled as pneumatic burst control for stabilization and orbital correction. This paper focuses on the utility of burst controls in shared control settings. Where traditional shared control strategies work in conjunction with or filter user input commands (see [8], [10], [14], [25], [31]), burst control strategies can provide for a different form of shared control. Potentially working along with existing shared controllers, burst control signals can be applied to implement quick changes in the trajectory on request, assisting with control goals as user / shared control tasks become too challenging to maintain. Due to their short duration, the key advantage of these signals is that they allow computers to take control authority and implement changes before users notice the loss in control. The challenge in implementing these on-demand, quick-assist controllers is that optimal control responses for nonlinear systems require constrained iterative optimization and so are infeasible to compute on-the-fly.

To address this concern, this paper introduces a method that allows for rapid generation of optimal burst control laws. First, the process computes the control action[2] to apply at any time to which a system's tracking objective is optimally sensitive. As Section II-A proves, this action is guaranteed to exist under standard assumptions. Planning for infinitesimal activation durations, we show optimal actions can be computed extremely rapidly from a simple algebraic expression of state and co-state (no iterative optimization). Next, Section II-B provides a simple line search that is guaranteed to find a finite application duration for any selected activation time, τ_m, that will provide a desired improvement in trajectory. By quickly computing optimal actions and resolving their finite application durations, the approach calculates assistive burst controls on-demand. Through interactive simulation, we demonstrate how this *control-on-request* (COR) paradigm can be implemented using simple interface components (e.g. buttons, toggles, or switches), providing users the ability to quickly request bursts of computer control assistance.

To illustrate how COR interfaces can complement traditional shared control systems, consider the scenario of a fighter pilot flying a jet. Modern pilots share control with sophisticated systems that provide low level commands to stabilize flight (see [20], [26], [27]). These control systems also serve as safety mechanisms, protecting against maneuvers that push the jet beyond recovery. However, to maneuver aggressively pilots require significant control authority and are still heavily relied on to actively stabilize flight. In these circumstances, if a pilot were to begin to lose control over a plane (e.g. entering a flat spin) a COR interface could prove extremely useful. While an optimal control response would likely take too long to compute, a pilot could activate the COR interface to momentarily increase computer control authority. Through a short burst of control, the computer may be able to improve the current trajectory enough to stabilize the plane or provide more time to respond.

Another use case for a COR controller is in rehabilitation. In this setting, patients recovering from stroke increasingly use powered assistive devices (e.g. Locomat [19], Ekso [33] and ALEX exoskeletons [29], etc.) under the supervision of

Alex Ansari and Todd Murphey are with the Department of Mechanical Engineering, McCormick School of Engineering and Applied Science, Northwestern University, 2145 Sheridan Road, Evanston, IL 60208, USA, alexanderansari2011@u.northwestern.edu, t-murphey@northwestern.edu

[1] For the purposes of this example, assume the toggles are body fixed to the ball such that users can apply control (strike the ball) at any time.

[2] In this paper a control action is a control vector with constant values applied for a (possibly infinitesimal) duration of time.

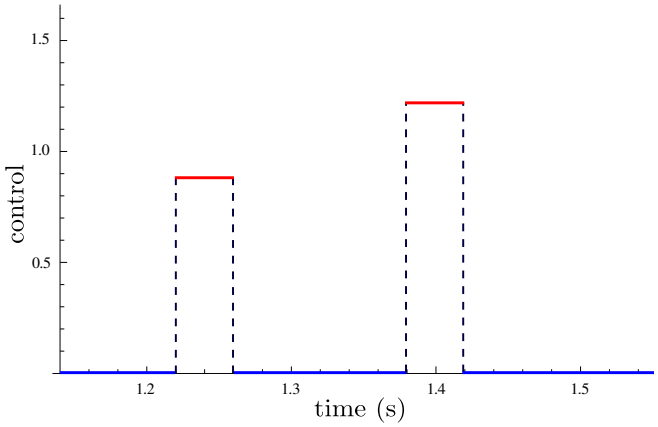


Fig. 1: An example burst control law. Continuous nominal control mode $u(t) = 0$ (blue curve) switches at $t = 1.22s$ and $t = 1.38s$ to alternate control modes (red) that are applied for short time horizons ($0.04s$).

physical therapists. These devices provide specified degrees of assistance by applying torques to knee and hip joints during treadmill-based gait training. However, research is focused on designing these devices to provide minimal assistance, encouraging patients to develop muscle and control systems while guiding motion and avoiding falls. Shared controllers vary the degree of assistance based on patient skill, but falling is still an issue and a safety harness is essential. In the scenario described, a monitoring physical therapist equipped with a COR interface could activate a response as a patient begins to fall. Using the techniques in this paper, an optimal burst control response could feasibly be computed quickly enough to stabilize the patient sufficiently for them to regain control and balance.

Following this Introduction, Section II derives a technique that takes advantage of the structure of burst control signals to compute an algebraic optimal control solution for systems with nontrivial dynamics. The methods produce control solutions that can be computed efficiently enough to enable the type of COR interfaces mentioned previously. To demonstrate the approach, Section III discusses a simple example application where a user is provided a 1-button COR interface to interactively request computer assistance to stabilize an unstable system. Section IV presents a discussion of this application with concluding remarks in Section V.

II. Burst Control Synthesis

This section presents a method to compute burst control laws that attain specified levels of improvement in trajectory tracking cost functionals. Section II-A leverages the short intended switching duration of burst control signals to avoid dynamic constraints and iterative optimization in solving for a schedule of infinitesimal optimal actions to apply at any time. These infinitesimal actions specify the magnitude and direction of the burst control. The following Section II-B provides a line search process that returns a finite duration to apply each action to produce a burst control law that achieves a desired reduction in tracking cost over a specified horizon.

A. Calculating a Schedule of Optimal Infinitesimal Actions

The type of systems addressed in this paper are assumed to follow continuous trajectories, $(x(t), u(t))$, such that

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

The dynamics vector, $f(x(t), u(t))$, can be nonlinear with respect to the state vector, $x(t) \in \mathbb{R}^n$, but is assumed to be linear (or has been linearized) with respect to control vector, $u(t) \in \mathbb{R}^m$. With these assumptions, (1) can be written in control-affine form

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

The goal of the proposed COR controllers is to predict the trajectory of the system and compute an optimal burst of control that produces a specified level of improvement in this trajectory when activated. To accomplish this, a cost functional is used to compare the performance of different trajectories. The state tracking cost functional,[3]

$$\begin{aligned} J_1 =& \int_{t_0}^{t_f} l_1(x(t), u(t))\,dt + m(x(t_f)) \\ =& \frac{1}{2}\int_{t_0}^{t_f} \|x(t) - x_d(t)\|_Q^2\,dt + \frac{1}{2}\,\|x(t_f) - x_d(t_f)\|_{P_1}^2\,, \end{aligned} \tag{3}$$

serves this purpose. For control derivations, J_1 need only obey general form (3). However, the quadratic form is applied in the implementation described in this paper. For this case, $Q = Q^T \geq 0$ defines the metric on incremental state tracking error and $P_1 = P_1^T \geq 0$ defines the metric on state error at terminal time t_f. Initial time, t_0, is assumed to be the activation time of the COR interface to select controls that improve the future system trajectory.

Assume the system can be in one of two dynamic modes at a given time that differ only in control. Under nominal conditions, the system applies control law, $u(t)$, and dynamics, $f_1 = f(x(t), u(t))$, result. Upon activating the COR interface, the dynamics switch from mode f_1 to an alternate dynamic mode $f_2 = f(x(t), u_2(t))$ for a short duration, λ^+, before switching back. Consider the extreme case where the system evolves according to f_1, and $u_2(t)$ is applied for an infinitesimal duration before switching back to the nominal control law, $u(t)$. In this case the mode insertion gradient,

$$\frac{dJ_1}{d\lambda^+} = \rho(t)^T[f_2 - f_1]\,, \tag{4}$$

measures the resulting change in cost functional (3) (see [11], [12], [28]). The term, $\rho(t) \in \mathbb{R}^n$, is the adjoint (co-state) variable calculated from the current system trajectory based on the differential equation,

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

such that at the final time $\rho(t_f) = D_x m(x(t_f))^T$. The control duration, λ^+, is evaluated infinitesimally, as $\lambda^+ \to 0$.

Through the course of the trajectory, any time, τ_m, the mode insertion gradient is negative, the state tracking cost

[3]The notation, $\|\cdot\|_M^2$, indicates a norm on the argument where matrix, M, provides the metric (i.e. $\|x(t)\|_Q^2 = x(t)^T\,Q\,x(t)$).

functional can be reduced if $u_2(\tau_m)$ is activated for some duration at that time. The magnitude of the mode insertion gradient provides a first-order model of the change in trajectory cost that would result relative to the duration of application of this control.

To produce a desired degree of improvement in a system trajectory, each applied burst of control authority needs to improve cost (3) by a specified amount. In other words, control actions, $u_2^*(\tau_m)$, need to be computed that drive the mode insertion gradient to a desired negative value, $\alpha_d \in \mathbb{R}^-$. However, there is generally a cost associated with the application of control authority. As such, a trade-off must be made in tracking the desired value of the mode insertion gradient, α_d, relative to control effort. Following a trajectory optimization approach, these competing goals can be encoded into the cost functional,

$$\begin{aligned} J_2 =& \int_{t_0}^{t_f} l_2(x(t), u(t), u_2(t), \rho(t))\, dt \\ =& \frac{1}{2}\int_{t_0}^{t_f} [\frac{dJ_1}{d\lambda^+} - \alpha_d]^2 + \|u_2(t)\|_R^2\, dt \\ =& \frac{1}{2}\int_{t_0}^{t_f} [\rho(t)^T(f_2 - f_1) - \alpha_d]^2 + \|u_2(t)\|_R^2\, dt\,. \end{aligned} \quad (6)$$

Matrix R allows the designer to encode the cost of control relative to tracking α_d. The continuous schedule[4] of control actions, $u_2^*(t)$, that minimizes (6), optimizes this trade-off.

The remainder of the paper assumes quadratic norms in (6) with $R = R^T > 0$. Proved in [4], quadratic functionals and the space of positive semi-definite / definite cones is convex. Because convexity is preserved for non-negative weighted sums and integration, this choice of $R > 0$ provides convexity of (6). Additionally, the nominal control is often (indicated by case) assumed to be null, $u(t) = 0$, over the control planning horizon. This choice allows easier interpretation of $u_2^*(\tau_m)$ as the optimal action at τ_m relative to doing nothing (allowing the system to drift for a horizon into the future).

Theorem 1: Define $\Lambda \triangleq h(x(t))^T\, \rho(t)\, \rho(t)^T\, h(x(t))$. The schedule of controls, $u_2(t)$, that optimizes (6) based on dynamics (2) and state tracking cost functional (3) is given by the algebraic expression,

$$u_2^*(t) = (\Lambda + R^T)^{-1}\, [\Lambda\, u(t) + h(x(t))^T \rho(t)\, \alpha_d]\,. \quad (7)$$

Proof: Any time $t \in [t_0, t_f]$, action $u_2(t)$ is assumed to be applied infinitesimally and so does not affect state trajectory (i.e. $x = x(u(t), t)$). Optimization of convex cost (6) with respect to $u_2(t)\, \forall t \in [t_0, t_f]$ is therefore unconstrained by (2) and it is necessary and sufficient for (global) optimality to find a curve $u_2(t)$ that sets its first variation to 0. Using the Gâteaux derivative and the definition of the functional derivative,

$$\begin{aligned} \delta J_2 =& \int_{t_0}^{t_f} \frac{\delta J_2}{\delta u_2(t)} \delta u_2(t)\, dt \\ =& \frac{d}{d\epsilon}\int_{t_0}^{t_f} l_2(x(t), u(t), u_2 + \epsilon\, \eta(t), \rho(t))\, dt|_{\epsilon=0} \\ =& \int_{t_0}^{t_f} \frac{d}{d\epsilon} l_2(x(t), u(t), u_2 + \epsilon\, \eta(t), \rho(t))|_{\epsilon=0}\, dt \\ =& \int_{t_0}^{t_f} \frac{\partial l_2(x(t), u(t), u_2(t), \rho(t))}{\partial u_2(t)}\, \eta(t)\, dt \\ =& 0\,, \end{aligned} \quad (8)$$

where ϵ is a scalar and $\epsilon\, \eta(t) = \delta u_2(t)$.

At the optimal value of $u_2(t)$ (i.e. $u_2(t) = u_2^*(t)$), the final equivalence in (8) must hold $\forall\, \eta(t)$. By the Fundamental Lemma of Variational Calculus (see [24]), this implies $\frac{\partial l_2(\cdot)}{\partial u_2(t)} = 0$ at the optimizer. The resulting expression,

$$\begin{aligned} \frac{\partial l_2(\cdot)}{\partial u_2(t)} =& (\rho(t)^T\, h(x(t))\, [u_2(t) - u(t)] - \alpha_d) \\ & \rho(t)^T\, h(x(t)) + \ u_2(t)^T\, R \\ =& h(x(t))^T\, \rho(t)\, (\rho(t)^T\, h(x(t)) \\ & [u_2(t) - u(t)]\ - \alpha_d) + R^T u_2(t) \\ =& [h(x(t))^T\, \rho(t)\, \rho(t)^T\, h(x(t))]\, u_2(t) \\ & + R^T u_2(t) - [h(x(t))^T\, \rho(t)\, \rho(t)^T\, h(x(t))] \\ & u(t) - h(x(t))^T\, \rho(t)\, \alpha_d = 0\,, \end{aligned} \quad (9)$$

can therefore be solved in terms of $u_2(t)$ to find the value, $u_2^*(t)$, that minimizes (6). Algebraic manipulation confirms this optimal value is given by (7). ■

Though the principles applied to derive the schedule of infinitesimal optimal control actions (7) are reasonable, they are also non-traditional. To provide intuition regarding the properties of these solutions, the following proposition proves that the optimization posed to derive these controls is equivalent to a well-studied class of Tikhonov regularization problems (see [7], [9], [13], [16]).

Proposition 1: Assume $u, u_2, \rho, h \in \mathcal{H}$ where $\mathcal{H}$ is an infinite dimensional reproducing kernel Hilbert function space (RKHS).[5] With appropriate change of variables, minimization of (6) obeys the structure of generalized continuous-time linear Tikhonov regularization problem [6]

$$\min_z \|\Gamma z - b\|^2 + \kappa^2 \|L(z - z_0)\|^2\,, \quad (10)$$

and (7) obeys the structure of associated solution

$$z^* = (\Gamma^T\Gamma + \kappa^2 L^T L)^{-1}(\Gamma^T b + \kappa^2 L^T L z_0)\,. \quad (11)$$

Above, Γ and L are bounded linear operators on $\mathcal{H}$, vectors z and $z_0 \in \mathcal{H}$, and b and $\kappa \in \mathbb{R}$. See [7], [9], [13], and [16] for more detail on (10) and (11).

[4] At any specified application time, τ_m, of the COR interface, $u_2^*(\tau_m)$ represents the optimal action that balances control authority and drives the mode insertion gradient to α_d if activated around that time. Thus, $u_2^*(t)$ is a schedule of optimal actions that can be switched to from nominal control, $u(t)$, to produce a desired change in mode insertion gradient at τ_m.

[5] Practically, this is not a very stringent requirement. Most spaces of interest are RKHS. Refer to [7] for more detail.

[6] For equivalence, $\|\cdot\|$ refers to the $\mathcal{L}_2$ norm and $\mathcal{H}$ is additionally assumed to be an $\mathcal{L}_2$ space.

Proof: Using the control affine form of dynamics f_1 and f_2, the final equality in (6) can be stated as

$$J_2 = \frac{1}{2}\int_{t_0}^{t_f} [\rho(t)^T h(x(t))(u_2(t)-u(t))-\alpha_d]^2+\|u_2(t)\|_R^2\, dt\,.$$

Performing change in variables $z(t) = u_2(t) - u(t)$, $z_0(t) = -u(t)$, $\Gamma = \rho(t)^T h(x(t))$, and $b = \alpha_d$ yields

$$J_2 = \frac{1}{2}\int_{t_0}^{t_f} [\Gamma z(t) - b]^2\, dt + \frac{1}{2}\int_{t_0}^{t_f} \|z(t) - z_0(t)\|_R^2\, dt\,.$$

Because $R = R^T > 0$, it can be Cholesky factorized as $R = M^T M$. By pulling out a scaling factor κ^2, the factorization can be rewritten $R = M^T M = \kappa^2(L^T L)$. Applying this factorization and posing the expression in terms of $\mathcal{L}_2$ norms results in

$$J_2 = \frac{1}{2}\|\Gamma z(t) - b\|^2 + \frac{1}{2}\|\kappa L(z(t) - z_0(t))\|^2\,.$$

Minimization of (6) is thus equivalent to (10) up to a constant factor of $\frac{1}{2}$ that can be dropped as it does not affect z^*.

Additionally, equivalence of solutions (7) and (11) can be proved directly. With the previous change of variables, $u_2^*(t)$ can be written as $z^*(t) - z_0(t)$ and (7) as

$$z^*(t) - z_0(t) = (\Gamma^T\Gamma + \kappa^2 L^T L)^{-1}(-\Gamma^T\Gamma z_0(t) + \Gamma^T b)\,.$$

Algebraic manipulation verifies this is equal to Tikhonov regularization solution (11). ■

As the following corollary indicates, because minimization of (6) can be posed as a Tikhonov regularization problem, solutions (7) inherit useful properties that regularization solutions obey.

Corollary 1: With the assumptions stated in Proposition 1, solutions (7) for minimization of (6) exist and are unique.

Proof: The proof follows from Proposition 1, which shows the minimization can be formulated as a Tikhonov regularization problem with convex $\mathcal{L}_2$ error norm, $\|\Gamma z - b\|$. These problems are guaranteed to have solutions that exist and are unique. A proof is provided in [9]. ■

Globally, optimal control actions (7) inherit properties of Tikhonov regularization solutions. However, the following corollary shows that near equilibrium points, solutions (7) simplify to linear state feedback laws.

Corollary 2: Assume system (2) contains equilibrium point $x = 0$, the state and co-state are continuous, and tracking cost (3) is quadratic[7]. There exists a neighborhood around the equilibrium and nonzero time horizon for which optimal actions (7) are equivalent to linear feedback regulators.

Proof: At final time, $\rho(t_f) = P_1 x(t_f)$. Due to continuity, this linear relationship between the state and co-state must exist for a nonzero neighborhood around t_f such that

$$\rho(t) = P(t)\, x(t)\,. \tag{12}$$

[7]A quadratic cost is assumed so that resulting equations emphasize the local similarity between burst control and LQR [3].

Applying this relationship, (7) can formulated as

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

This expression contains terms quadratic in $x(t)$. In the neighborhood of the equilibrium these quadratic terms go to zero faster than the linear terms, and controls converge to

$$u_2^*(t) = R^{-T} h(x(t))^T\, P(t)\, x(t)\, \alpha_d\,. \tag{13}$$

By continuity, in a sufficiently small neighborhood of the equilibrium the system dynamics can be approximated as LTV system, $\dot{x}(t) = A(t)\, x(t) + B(t)\, u(t)$, (where $A(t)$ and $B(t)$ are linearizations about the equilibrium). Applying this assumption and differentiating (12) produces

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

Inserting relation (5) yields

$$\begin{aligned} -D_x l_1(\cdot)^T - A(t)^T P(t)\, x(t) = \dot{P}(t)\, x(t)\; +\; P(t) \\ (A(t)\, x(t) + B(t)\, u(t))\,, \end{aligned}$$

which can be re-arranged such that

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

When nominal control $u(t) = 0$, this reduces to

$$0 = Q + A(t)^T P(t) + P(t)\, A(t) + \dot{P}(t). \tag{14}$$

Note the similarity to a Lyapunov equation. As mentioned, this relationship must exist for some nonzero neighborhood of t_f. Therefore, by continuity of $\rho(t)$, there must exist a finite time horizon and neighborhood of the equilibrium where (7) simplifies to linear feedback regulator (13) and $P(t)$ can be computed from (14) subject to $P(t_f) = P_1$. ■

As in model predictive control (MPC) from [1], [2], [6], [15], it is possible to compute open-loop optimal actions (in this case $u_2^*(\tau_m)$) to provide finite-horizon tracking improvements and to sequence these in closed-loop. This would be equivalent to continuously activating a COR interface. In such implementations, one could specify α_d to provide local stability based on (13). Alternatively, if (3) is quadratic and nominal control $u(t)$ modeled as applying consecutively computed optimal actions (13) near equilibrium, (14) becomes a Riccati differential equation for the closed-loop system (see [17]) and actions (13) become finite horizon LQR controls [3]. In this case one can prove the existence of a Lyapunov function and guarantee stability using methods from MPC and LQR theory to drive $\dot{P}(t) \to 0$ ([1], [15], [17], [18], [21], [23]). While beyond this scope, we have begun to explore close-loop implementation and stability to leverage the efficient synthesis methods presented.

B. Computing the Control Duration

Theorem 1 provides a means to compute a schedule of open-loop optimal control actions, $u_2^*(t)$. When implemented infinitesimally around any time, τ_m, $u_2^*(\tau_m)$ is the needle variation (see [32]) in $u(\tau_m)$ that optimizes control authority in driving the mode insertion gradient, $\frac{dJ_1}{d\lambda^+}$, to α_d at that time. This value of the mode insertion gradient reflects the achievable sensitivity of cost (3) to application of $u_2^*(\tau_m)$ for infinitesimal duration. However, by continuity of the adjoint and mode insertion gradient as $\lambda^+ \to 0$, there exists an open, non-zero neighborhood, V, around $\lambda^+ \to 0$ for which the mode insertion gradient models this sensitivity to first order (see [5], [12]). Hence the mode insertion gradient can be used to model the change in cost achievable by application of $u_2^*(\tau_m)$ over a finite duration $\lambda^+ \in V$ as

$$\Delta J_1 \approx \frac{dJ_1}{d\lambda^+}\bigg|_{\lambda^+ \to 0} \lambda^+. \tag{15}$$

As $u_2^*(\tau_m)$ regulates $\frac{dJ_1}{d\lambda^+} \approx \alpha_d$, (15) becomes $\Delta J_1 \approx \alpha_d \lambda^+$. Thus the choice of λ^+ and α_d allow the control designer to specify the desired degree of change in (3) provided by each $u_2^*(\tau_m)$. Also note that for a given $\frac{dJ_1}{d\lambda^+}$ and any choice of λ^+, (15) can be applied to test if $\lambda^+ \in V$ or that it at least provides a ΔJ_1 that is known to be achievable for $\lambda^+ \in V$.

Though the neighborhood where (15) provides a reasonable approximation varies depending on system, in practice it is fairly straightforward to select a $\lambda^+ \in V$. The easiest approach is to select a single conservative estimate for λ^+. This is analogous to choosing a small, fixed time step in finite differencing or Euler integration. However, to avoid a-priori selection of a $\lambda^+ \in V$ and unnecessarily conservative step sizes, we use a line search with a descent condition to select a $\lambda^+ \in V$ or one that provides a minimum change in cost (3). Described in [30], the line search iteratively reduces the duration from a default value. By continuity, the process is guaranteed to find a duration that produces a change in cost within tolerance of that predicted from (15). In implementation, we use an iteration limit to bound the algorithm in time. Note that failing to find an acceptable λ^+ within the iteration limit is not usually a concern because (7) provides a schedule of control values so a duration can be sought for any nearby time if the system is overly sensitive at the current time.

Finally, because the pair (α_d, λ^+) determines the change in cost that a control can provide, it is worth noting that a sufficient decrease condition similar to the one proposed in [5] can be applied during the line search. In addition to the methods proposed in Section II-A, stability in closed-loop, MPC style implementation can be provided by defining this condition to guarantee each control provides a sufficient reduction in cost to avoid convergence to suboptimal states.

Algorithm 1 Burst Control Synthesis

Initialize α_d, minimum change in cost ΔJ_{min} from (15), current time t_{curr}, default control duration Δt_{init}, scale factor $\beta \in (0,1)$, time horizon T, and the max line search iterations i_{max}.
if COR interface activated **then**
  $(t_0, t_f) = (t_{curr}, t_{curr} + T)$
  Simulate $(x(t), \rho(t))$ for $t \in [t_0, t_f]$ from f_1
  Compute initial cost $J_{1,init}$
  Specify α_d
  Compute $u_2^*(t)$ from $(x(t), \rho(t))$ using Theorem 1
  $\tau_m = \frac{\Delta t_{init}}{2} + t_0$
  Initialize $i = 0$, $J_{1,new} = \infty$
  while $J_{1,new} - J_{1,init} > \Delta J_{min}$ **and** $i \leq i_{max}$ **do**
    $\lambda^+ = \beta^i \times \Delta t_{init}$
    $(\tau_0, \tau_f) = (\tau_m - \frac{\lambda^+}{2}, \tau_m + \frac{\lambda^+}{2})$
    Re-simulate $x(t)$ applying f_2 for $t \in [\tau_0, \tau_f]$
    Compute new cost $J_{1,new}$
    $i = i + 1$
  end while
end if
return $(u_2^*(\tau_m), \tau_0, \tau_f)$

Algorithm 1: The algorithm above includes an optional line search phase which re-simulates the state $x(t)$ and trajectory cost until an appropriate duration for application of $u_2^*(\tau_m)$ can be found. This is only required if the approximation provided by the mode insertion gradient is not accurate over the default control interval chosen.

III. COR Interface Example: Interactive Stabilization of an Unstable System

This section presents a simple, interactive simulation created to demonstrate a COR interface in a real-time control scenario. The goal of the demonstration is for users to stabilize an unstable system using nothing but a single button COR interface to request bursts of computer assistance. To highlight the impact of the interface, the nominal control is $u(t) = 0$, so that the system follows an unstable spiral trajectory, increasing in speed as it moves away from the origin. The COR interface is designed to help stabilize the system from any state by driving it toward the origin.

For the system discussed, the state, $x(t)$, consists of $2D$ position components, $(x_1(t), x_2(t))$, and the control contains velocity inputs, $(\dot{x}_1(t), \dot{x}_2(t))$. The evolution is described by unstable dynamics, $f(x(t), u(t)) = A\,x(t) + B\,u(t)$, with

$$A = \begin{pmatrix} 0.1 & 0.1 \\ -10 & 0.1 \end{pmatrix} \text{ and } B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{16}$$

The goal for the COR interface is encoded by setting the desired trajectory, $x_d(t)$, from (3) to the origin, with $Q = I$.

The COR interface is designed to apply an optimal action from the schedule, $u_2^*(t)$, from Theorem 1 for a short duration such that a desired improvement in trajectory cost functional (3) results. To accomplish this, α_d is chosen so that the mode insertion gradient (sensitivity of the trajectory cost functional to activation of $u_2^*(\tau_m)\ \forall \tau_m$) is negative. To select control laws that produce significant reductions in cost, α_d is set to -1000. The R matrix from (6) that weights the

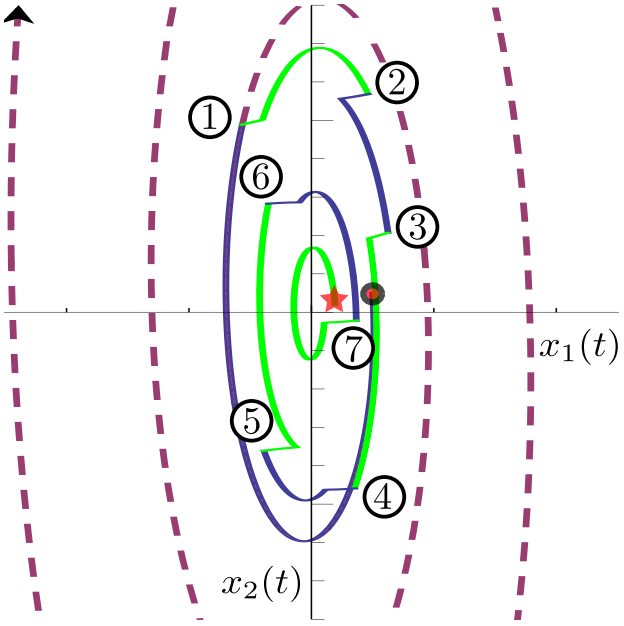


Fig. 2: A user presses a 1-button COR interface to activate short control responses at different points along the trajectory of a system with unstable dynamics. Each activation is marked consecutively by number. The system starts at the black and red circle and follows a nominal trajectory until the COR interface is first used. At this point, the actual trajectory diverges from the nominal trajectory (dashed purple) that would result if COR responses were never applied. Upon each activation, the trajectory color switches between blue and green to highlight the transition. The COR interface calculates and applies each $u_2^*(\tau_m)$ for short duration, producing a stabilizing effect that drives the system toward the origin as desired. The final position is indicated with a star.

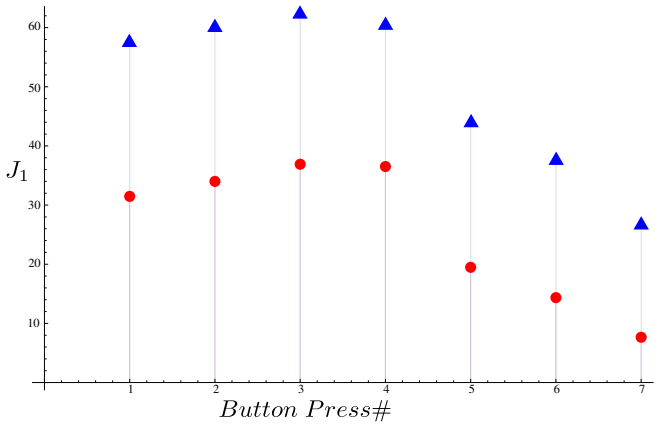


Fig. 3: Changes in cost functional (3) resulting due to multiple activations of the COR interface over a typical trajectory. Costs are computed based on the $4s$ trajectory segment following each button press that occurs over a $20s$ period. Blue triangles indicate the original trajectory cost before application of $u_2^*(\tau_m)$ and red circles indicate the cost after. At each press, the burst control applied by the COR interface reliably improve trajectory costs.

norm on applied controls is initialized to the identity matrix. However, this matrix can be modified by the user to select for control laws that use more or less control authority, resulting in better or worse tracking of α_d, respectively.

Another design choice in creating the COR interface is the selection of the time horizon for integration of these cost functionals. This time-frame specifies how long control $u_2^*(\tau_m)$ has to affect the desired change in system trajectory. For the same change in trajectory, short time horizons can result in significant control effort. Too long a time horizon and $u_2^*(\tau_m)$ will only produce minor changes in trajectory, as these accumulate to significantly differences in the cost functional over time. The COR interface used to control the simulated system in (16) uses a $4s$ time horizon to balance control authority and produce a noticeable stabilizing effect.

For interactive implementation, the system is forward simulated for $4s$ and the resulting trajectory animated in real-time. As the trajectory nears the end of the time horizon, it is re-simulated from the current state for another $4s$ so that it evolves continuously. If the COR interface button is pressed, the trajectory is immediately forward simulated from the current state for another $4s$ so that when control $u_2^*(\tau_m)$ is applied, the resulting change in trajectory will be produced over a $4s$ time horizon from the current time. Using this new trajectory, $u_2^*(t)$ is computed from (7).

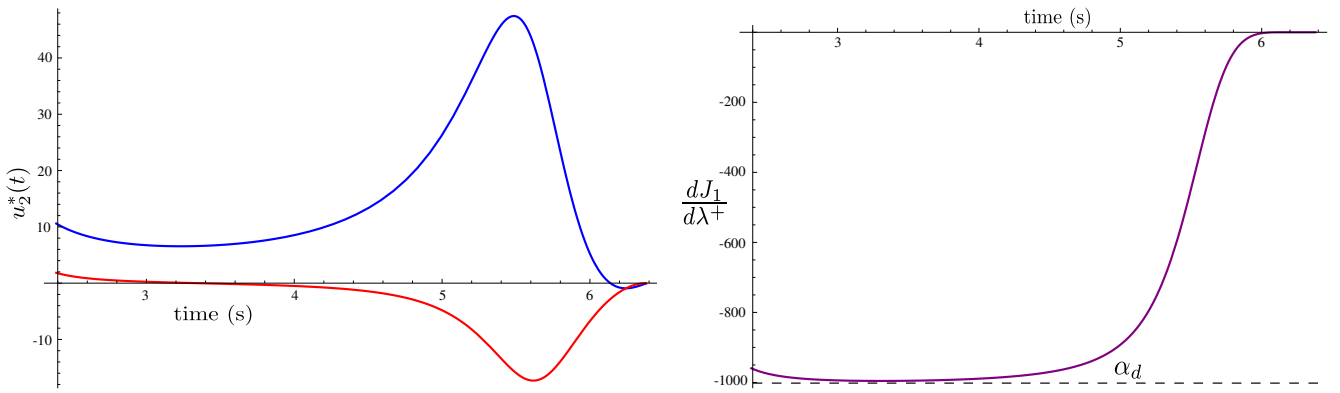
Algorithm 1 is applied to determine an appropriate duration for application of $u_2^*(\tau_m)$, where τ_m is set to the current time plus half the default duration of $0.03s$.[8] Starting with this default duration, the trajectory produced from application of $u_2^*(\tau_m)$ is simulated $4s$ into the future and the change in cost (3) is computed based on the original and new trajectories. If simulated application of $u_2^*(\tau_m)$ does not result in a specified minimum change in cost functional the control duration is reduced, the trajectory is re-simulated, and the process repeated until an appropriate duration can be found. (In practice, this system rarely required reduction of the default duration.) As Figs. 2 and 3 show, the process produces regular changes in trajectory and provides clear improvement, moving the system toward the origin.

IV. Results and Discussion

The interactive example from Section III illustrates the efficiency of the methods discussed in this paper. The demonstration, implemented in *Mathematica*, runs in real-time on a laptop with an Intel Core 2 Duo chipset. Based on these results an optimized version implemented in an appropriately efficient language (e.g. C) can be expected to enable embedded implementation of COR interfaces for application in the time critical scenarios discussed.

The snapshot in Fig. 2 shows a typical trajectory that results after $t \approx 20s$. In this case, the user presses the COR interface button multiple times to request compute aid in improving the system trajectory. The dashed curve shows

[8] A default duration of $0.03s$ is applied because it provided predictable trajectory improvements that users found easy to control.



(a) (b)

Fig. 4: The schedule of optimal actions $u_2^*(t)$ computed based on a $4s$ segment of the system trajectory (Fig. 4a). The blue and red curves represent velocity controls $\dot{x}_1(t)$ and $\dot{x}_2(t)$ (ft/s), respectively. At all times, actions minimize control authority while driving the mode insertion gradient toward $\alpha_d = -1000$ at that time according to weighting matrices Q and R. The mode insertion gradient curve (Fig. 4b) provides a first order approximation of how the trajectory tracking cost functional would change due to a short duration switch from nominal control $u(t) = 0$ to $u_2^*(t)$ at different times along the trajectory. In this case, one can infer that the control action should be applied within $2.5s$, before the mode insertion gradient becomes too small. Regardless of whether the system obeys linear dynamics or nonlinear dynamics (2), the procedure for calculating the mode insertion gradient and optimal actions remains unchanged.

the trajectory that would have resulted if the interface were never used. At each point where the button is pressed a similar change in trajectory results. No matter when along the trajectory the button is pressed, the COR interface brings the system closer to the origin.

Similarly, Fig. 3 demonstrates the cost reductions obtained using the COR interface multiple times over the course of a typical trajectory. Each time the interface is activated, the cost for the next $4s$ trajectory segment is simulated both before (blue triangles) and after (red circles) application of $u_2^*(\tau_m)$. The burst control generated by the COR interface produces a significant reduction in cost after each activation. As illustrated, trajectory costs are improved by the specified minimum degree of $\geq 5\%$ of the current cost in all cases.

Fig. 4 shows a typical schedule of control actions, $u_2^*(t)$, and the mode insertion gradient curve indicates the sensitivity of (3) to a switch from control $u(t) = 0$ to $u_2^*(t)$ at each point in time along the trajectory. As mentioned, when the button is pressed the trajectory is re-simulated $4s$ into the future and $u_2^*(t)$ is computed. In this case the COR interface button was pressed at $t \approx 2.4s$ and each plot covers a $4s$ time horizon from this initial time. The short duration switch to $u_2^*(\tau_m = t + \lambda^+/2)$ is designed to occur when the button is pressed and lasts for the default duration of $\lambda^+ \approx 0.03s$ (unless the line search reduces this duration). Based on the mode insertion gradient, switching to the control $u_2^*(\tau_m) \approx (10ft/s, 2ft/s)$ at this time will change the trajectory cost functional at a rate of $\approx \frac{-960}{s}$. Applying the control for $0.03s$ can thus result in a change $\Delta J_1 \approx -29$ over the course of $4s$.

Because control schedule $u_2^*(t)$ is continuous for each $4s$ trajectory simulation, it can be applied at any time rather than at (near) the initial time. However, as shown in Fig. 4a, control effort increases farther along the trajectory before dropping to 0. This is because more control authority is required to achieve the same change in cost functional toward the end of the trajectory. In other words, if the same change in cost of $\Delta J_1 \approx -29$ is desired, but the switch to $u_2^*(\tau_m)$ occurs $3s$ into the $4s$ trajectory ($\tau_m \approx 5.42s$) rather than at the beginning, much more control effort will be required to achieve that desired change in the $1s$ remaining. This is also why both $u_2^*(t)$ and the mode insertion gradient become 0 at the end of the trajectory. At the final time, no amount of control authority can improve cost functional (3). However, by decreasing the weights in R, $u_2^*(t)$ can be allowed to apply more control effort and the mode insertion gradient will better approximate a flat line at α_d. Both terms will still drop off steeply near the end of the trajectory.

Due to limitations on available control authority, it is possible that there are additional times along the trajectory where no control can improve the cost. The mode insertion gradient naturally indicates these cases by dropping to 0. However, a COR interface is still useful in such circumstances because $u_2^*(t)$ can applied at any time. To deal with this issue, the curve of $u_2^*(t)$ can be searched to find a nearby time where control authority once again proves useful, and activation held until then. This change requires minor modification and is extremely useful for systems whose trajectories pass through areas in the null space of their controls. For example, for a cart-pendulum system where acceleration of the cart is the control, no amount of control authority can help swing up the pendulum if activated when the pendulum is horizontal. By waiting for the pendulum to swing beyond this point, the trajectory can once again be improved by available controls.

V. Conclusion

This paper presents an algebraic expression to compute optimal actions for nonlinear systems at any time. We demonstrate how a line search can resolve short durations to apply these actions to provide long time horizon improvement in tracking. Compared to standard methods where optimal controls are synthesized based on finite switching durations, these methods completely avoid constrained iterative optimization. They facilitate the rapid exchange between human and computer control required to enable a new shared control paradigm where automated assistance can be provided quickly and on-demand.

Interactive simulation results show this *control-on-request* (COR) paradigm can request, compute, and activate bursts of computer assistance in real-time. Due to space limitations, the example depicted is simple and intended only to demonstrate presented concepts. For more involved implementation examples where COR interfaces provide assistance to humans in balancing and rehabilitation tasks involving exoskeletons see [22]. Finally, due to promising preliminary results, we recommend closed-loop implementation of the described methods for future investigation.

ACKNOWLEDGMENT

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

References

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

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

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

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

[5] TM Caldwell and TD Murphey. Projection-based optimal mode scheduling. In *IEEE Conference on Decision and Control (CDC)*, 2013.

[6] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer, 2013.

[7] Zhe Chen and Simon Haykin. On different facets of regularization theory. *Neural Computation*, 14(12):2791–2846, 2002.

[8] Christian Cipriani, Franco Zaccone, Silvestro Micera, and Maria Chiara Carrozza. On the shared control of an emg-controlled prosthetic hand: analysis of user–prosthesis interaction. *IEEE Transactions on Robotics*, 24(1):170–184, 2008.

[9] Ernesto De Vito, Lorenzo Rosasco, Andrea Caponnetto, Michele Piana, and Alessandro Verri. Some properties of regularized kernel methods. *The Journal of Machine Learning Research*, 5:1363–1390, 2004.

[10] Eric Demeester, Alexander Hüntemann, Dirk Vanhooydonck, Gerolf Vanacker, Hendrik Van Brussel, and Marnix Nuttin. User-adapted plan recognition and user-adapted shared control: A bayesian approach to semi-autonomous wheelchair driving. *Autonomous Robots*, 24(2):193–211, 2008.

[11] Magnus Egerstedt, Yorai Wardi, and Henrik Axelsson. Optimal control of switching times in hybrid systems. In *9th IEEE International Conference on Methods and Models in Automation and Robotics*, 2003.

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

[13] Joel N Franklin. Minimum principles for ill-posed problems. *SIAM Journal on Mathematical Analysis*, 9(4):638–650, 1978.

[14] Paul Griffiths and R Brent Gillespie. Shared control between human and machine: Haptic display of automation during manual control of vehicle heading. In *Proceedings 12th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 358–366. IEEE, 2004.

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

[16] Per Christian Hansen. *The L-curve and its use in the numerical treatment of inverse problems*. IMM, Department of Mathematical Modelling, Technical Universityof Denmark, 1999.

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

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

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

[20] A Kreiner and K Lietzau. The use of onboard real-time models for jet engine control. *MTU Aero Engines, Germany*, 2003.

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

[22] Anastasia Mavrommati, Alex Ansari, and Todd Murphey. Optimal control-on-request: An application in real-time assistive balance control. In *IEEE International Conference on Robotics and Automation (ICRA).*, 2015.

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

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

[25] J. Philips, J. del R.Millan, G. Vanacker, E. Lew, F. Galan, P.W. Ferrez, H. Van Brussel, and M. Nuttin. Adaptive shared control of a brain-actuated simulated wheelchair. In *IEEE 10th International Conference on Rehabilitation Robotics (ICORR)*, pages 408–414, 2007.

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

[27] Robert M Taylor, Lex Brown, and Blair Dickson. From safety net to augmented cognition: Using flexible autonomy levels for online cognitive assistance and automation. Technical report, DTIC Document, 2003.

[28] Yorai Wardi and Magnus Egerstedt. Algorithm for optimal mode scheduling in switched systems. In *American Control Conference*, pages 4546–4551, 2012.

[29] Kyle N Winfree, Paul Stegall, and Sunil K Agrawal. Design of a minimally constraining, passively supported gait training exoskeleton: ALEX II. In *IEEE International Conference on Rehabilitation Robotics (ICORR)*, pages 1–6. IEEE, 2011.

[30] SJ Wright and J Nocedal. *Numerical optimization*, volume 2. Springer New York, 1999.

[31] Haoyong Yu, Matthew Spenko, and Steven Dubowsky. An adaptive shared control system for an intelligent mobility aid for the elderly. *Autonomous Robots*, 15(1):53–66, 2003.

[32] Jerzy Zabczyk. *Mathematical control theory: an introduction*. Springer, 2009.

[33] Adam B Zoss, Hami Kazerooni, and Andrew Chu. Biomechanical design of the berkeley lower extremity exoskeleton (BLEEX). *IEEE/ASME Transactions on Mechatronics*, 11(2):128–138, 2006.