

Characteristics of Human Pointing Motions with Acceleration*

Paul Varnell¹ and Fumin Zhang²

Abstract—Pointer acceleration is often applied to computer mice and other types of pointing interfaces to increase the effective speed and range of pointing motions without sacrificing precision during slow movements. However, the effects of pointer acceleration are not well understood and have been shown to be detrimental in some cases. We approach the problem of analyzing the effects of pointer acceleration from a systems perspective and show some general properties of a new dynamics model for pointer acceleration. Specifically, using our new model for pointer acceleration, along with the existing VITE model of human pointing motions, we show that the closed loop pointing system is stable for any acceleration profile, assuming there are no feedback delays or perturbations. We additionally show some state and performance bounds for this case, which could potentially be used to design pointing interfaces with acceleration.

I. INTRODUCTION

In human pointing interfaces, such as computer mice and joysticks, it is often desirable that users both be able to perform fast, long-range movements in a limited workspace with minimal required effort and be able to perform slow, precise movements for making corrections. Pointer acceleration can be used to achieve these dual goals by increasing the pointer sensitivity as the pointer speed increases, and thus creating a much wider range of pointer speeds. The benefits of pointer acceleration have caused it to be adopted as the default in most major operating system mouse interfaces [1]–[3] and in many other pointing interfaces.

While pointer acceleration is widespread, there is no consensus on what is the right type or amount of pointer acceleration to use for any given interface or task. Real implementations of pointer acceleration are generally designed by manual tweaking and experimenting with different acceleration profiles without using clear design principles. Furthermore, there are some potential downsides to pointer acceleration that are not completely understood. One commonly cited example [3] is that pointer acceleration may make it more difficult for the user to predict the pointer's motion, which may decrease pointing speed and accuracy, decrease the user's ability to produce desired motions in a repeatable way, and worsen the user's subjective rating of the feel of the device. We wish to more rigorously understand the effects of pointer acceleration, so we may be able to design pointing interfaces which keep its positive effects and reduce its negative effects.

*The research work is supported by ONR grants N00014-10-10712 (YIP) and N00014-14-1-0635; and NSF grants OCE-1032285, IIS-1319874, and CMMI-1436284.

¹Georgia Institute of Technology paul.varnell@gatech.edu

²Georgia Institute of Technology fumin@gatech.edu

We approach the problem of analyzing the effects of pointer acceleration from a systems perspective. We formulate a dynamics model for pointer acceleration and analyze its stability and performance properties when connected to an existing model for human pointing dynamics. These results may help shed light on the impact of pointer acceleration on pointer interfaces and could in the future be used to better design such interfaces.

Human pointing motions have been extensively studied in the fields of psychology, physiology, neuroscience, and human-computer interaction [4]. Particularly relevant to our work are the famous Fitts's law [5], which captures an invariant in pointing performance that applies for many different human pointing interfaces and tasks, and the VITE human pointing dynamics models [6], which describes pointing motions and reproduces Fitts's law. Several different papers [7]–[10] analyze the dynamics of human pointing as described by the VITE model, with and without feedback delays. In an earlier work [11], we showed how dissipativity theory could be used to analyze and design systems with delays, and we provided some preliminary analysis of the VITE pointing model in this framework.

Our research is novel in that, to the best of our knowledge, we are the first to consider pointer acceleration from a systems perspective. We present a new model of pointer acceleration that captures most existing implementations thereof, and we show several different general properties of this model. Specifically, we show that when the VITE human pointing model is connected in feedback to the model for pointer acceleration, the system is stable for any acceleration profile assuming there are no feedback delays or perturbations. We also show some bounds on the performance of the closed loop pointing system with acceleration assuming no delays or perturbations. We use simulation to illustrate and verify our claims. These results are significant for the analysis and design of pointing interfaces and provide a framework for new research in this area.

The remainder of the paper is organized as follows: in Section II we give some background on the dynamics of human pointing motions and the VITE human pointing model; in Section III we present a model of pointer acceleration and show the closed loop dynamics for human pointing with acceleration; in Section IV we prove that the closed loop human pointing dynamics are stable; in Section V we derive state bounds and performance bounds for the closed loop pointing system; in Section VI we show simulation results; and finally in Section VII we provide some closing remarks and directions for future research.

II. BACKGROUND: HUMAN POINTING DYNAMICS

Humans have been shown to generate similar motions when reaching and pointing with their arms, mouse pointers, laser pointers, and other devices [4]. We use the Vector Integration to Endpoint (VITE) model, as first shown in [6], to model pointing behaviors. This model can be written as

$$\begin{aligned}\dot{\eta} &= \gamma(-\eta + \rho - u) \\ \dot{y} &= g(t)[\eta]_d^+ ,\end{aligned}\quad (1)$$

where $y \in \mathbb{R}^n$ is the true position of the pointer specified by the user, $u \in \mathbb{R}^n$ is the feedback of the (perceived) position of the pointer, $\rho \in \mathbb{R}^n$ is the target pointer position, $\eta \in \mathbb{R}^n$ is a state called the difference vector, $g(t) \in \mathbb{R}_{\geq 0}$ is a gain called the go signal, and $\gamma \in \mathbb{R}_{\geq 0}$ is an internal system parameter. The operator $[\cdot]_d^+$ is used to switch the pointer motion off when the pointer overshoots the target. It is defined as

$$[\eta]_d^+ = \begin{cases} \eta, & \text{if } \langle \eta, d \rangle \geq 0 \\ 0, & \text{otherwise} \end{cases}, \quad (2)$$

where the vector $d \in \mathbb{R}^n$ will typically be defined as the direction from the pointer to the target at the initial time $d = \rho(0) - u(0)$. Note that the notation $[\eta]_d^+$ is a minor extension of the original VITE model that allows for arbitrary target locations.

The VITE model describes a single motion made by a human trying to drive the true position of the pointer y to the target pointer position ρ . If there is overshoot, and the true pointer position y passes over the target pointer position ρ , the human stops quickly, but no corrective action is taken to move the pointer back closer to the target. The parameters of this model are adjusted by the human to meet different kinds of performance objectives and can be thought of as very slowly time-varying. In Section V, we will see in more detail how these parameter choices affect performance.

III. POINTER ACCELERATION DYNAMICS

A. Dynamics Model

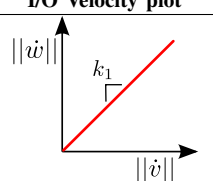
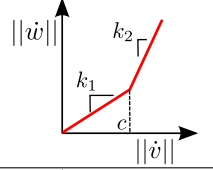
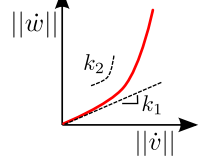
The basic design of pointer acceleration is to make the pointer more sensitive at higher speeds than at lower speeds. This allows the user to make quick and long pointer motions while still being able to make slow and precise pointer motions. The downside to pointer acceleration is that it may make it more difficult for the user to predict the pointer's motion and to reproduce precise motions. Furthermore, pointer acceleration can cause the perceived pointer position to drift away from the pointer position controlled by the user, which can cause problems in limited workspaces if that offset cannot be reset. Whether pointer acceleration improves the subjective feel of the pointing interface seems to be a very personal choice. Many operating systems turn on some form of pointer acceleration by default, while many modern video games that require precise pointer movement do not use acceleration.

There are a variety of different implementations of pointer acceleration, but most of them can be described by the following model. We can think of pointer acceleration as a transformation applied to the pointer position output of the human operator. If $v \in \mathbb{R}^n$ is the raw (unaccelerated) pointer position as specified by the user, then the accelerated output position $w \in \mathbb{R}^n$ satisfies

$$\dot{w} = G(\|\dot{v}\|) \dot{v}, \quad (3)$$

where $G: \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ is a non-decreasing scaling function, which is also referred to as the acceleration profile. Table I summarizes some of the most popular choices for the scaling function G , and Table II lists which operating systems use what scaling functions.

TABLE I: Common Pointer Acceleration Profiles

Name	Scaling Function	I/O Velocity plot
No Accel.	$G(\ \dot{v}\) = k_1$	
Threshold	$G(\ \dot{v}\) = \begin{cases} k_1, & \text{if } 0 \leq \ \dot{v}\ < c \\ k_2, & \text{if } \ \dot{v}\ \geq c \end{cases}$	
Linear	$G(\ \dot{v}\) = k_1 + k_2\ \dot{v}\ $	

Above, $k_2 \geq k_1$ are scaling constants and c is a speed threshold.

TABLE II: Operating Systems Acceleration Profiles

Implementation	Scaling Function
Microsoft Windows	Threshold (3 levels) [1], [12], [13]
Mac OSX	Linear [2] (Note 1)
Linux (xinput)	8 types including polynomial, linear, and threshold [3]

Note 1: This assessment is based only on visual inspection of the preference panel, as documentation of the acceleration profile could not be found.

We can see that the name ‘‘pointer acceleration’’ is actually somewhat of a misnomer, as most implementations do not involve acceleration at all, and are instead scaling the pointer’s velocity; although admittedly ‘‘pointer velocity scaling’’ is a less catchy title. One notable exception to this that does involve acceleration in some way, is the default ‘‘polynomial’’ scaling function referred to in [3], which can be represented as

$$G_{\text{polynomial}}(\|\dot{v}\|, \|\ddot{v}\|) = k_1 + k_2\|\dot{v}\|\|\ddot{v}\|$$

and uses the pointer input acceleration as well as the velocity to adjust the sensitivity. In this paper, we will not consider

scaling functions of this form, but it may be possible to extend our analysis to them in the future.

One other detail to note is since most pointing interface implementations only measure the pointer position and do not directly measure the pointer velocity, they must compute the pointer velocity via discretization so it can be used in the pointer acceleration system. This is normally not a significant issue, but if the pointer is moving very fast or the velocity scaling is large relative to the sampling rate, then the resulting accelerated pointer output produced by sampling may be significantly different from the ideal value. This is typically mitigated using filtering and by limiting the maximum pointer speed output from the acceleration system. We do not further consider the effects of pointer velocity sampling in this paper, but these effects could be modeled as a perturbation term added into the following closed loop pointing model.

B. Closed Loop Dynamics Model

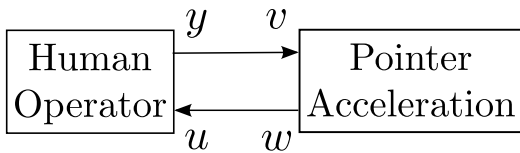


Fig. 1: Block diagram of closed loop pointing model.

We can connect the VITE model (1) with the pointing acceleration model (3) to create a closed loop pointing model. We make a few assumptions that simplify the following analysis. Our first assumption is there are no feedback delays or perturbations, so the closed loop system can be formed with feedback connection $u = w$ and $v = y$, as in Figure 1, and the overall state can be defined as $x = [w \ \eta]^\top$. Given this assumption, the closed loop pointing dynamics become

$$\begin{aligned} \dot{x} &= \begin{bmatrix} \dot{w} \\ \dot{\eta} \end{bmatrix} = \begin{bmatrix} G(|\dot{v}|) \dot{v} \\ \gamma(-\eta + \rho - u) \end{bmatrix} = \begin{bmatrix} G(|\dot{y}|) \dot{y} \\ \gamma(-\eta + \rho - w) \end{bmatrix} \\ &= \begin{bmatrix} G(|g(t)[\eta]_d^+|) g(t)[\eta]_d^+ \\ \gamma(-\eta + \rho - w) \end{bmatrix} \\ &= \begin{bmatrix} G(|g(t)[x_2]_d^+|) g(t)[x_2]_d^+ \\ \gamma(-x_2 + \rho - x_1) \end{bmatrix} . \end{aligned}$$

The effect of delays and perturbations on this model will be considered in future work, as it significantly complicates the analysis.

Secondly, we only consider one dimensional pointing motions $n = 1$. The original VITE model is also only one dimensional, and while most real pointing interfaces are two or three dimensional, the majority of the pointer motion is usually restricted to the line between the starting pointer position $x_1(0)$ and the target pointer position ρ . Some scenarios may necessitate modeling the additional pointing directions, such as when the target position is changing or when there is uncertainty in the pointer or target position, but in many cases it should suffice to model pointing motions as one dimensional.

Our third assumption is the go signal $g(t)$ is constant. Several authors using the VITE model only consider constant g , and this work may be able to be extended in the future to include the time-varying case.

Finally, without loss of generality, we assume the target position is located at the origin $\rho = 0$ and the starting pointer position is negative $x_1(0) < 0$. In general, this can be achieved by using a coordinate frame transformation.

Given these additional assumptions, the closed loop pointing dynamics become

$$\dot{x} = \begin{bmatrix} \widehat{G}(x_2) x_2 \\ -\gamma(x_1 + x_2) \end{bmatrix} = \widehat{A}(x) x \quad , \quad (4)$$

where $\widehat{G}(x_2) = g G(gx_2^+) I_{\{x_2 \geq 0\}}$, $x_2^+ = \begin{cases} x_2, & \text{if } x_2 \geq 0 \\ 0, & \text{else} \end{cases}$,

I is the indicator function, $\widehat{A}(x) = A(\widehat{G}(x_2))$, and $A(k) = \begin{bmatrix} 0 & k \\ -\gamma & -\gamma \end{bmatrix}$.

IV. STABILITY OF CLOSED LOOP DYNAMICS

We will now show the stability of the closed loop pointing dynamics (4) using Lyapunov-based methods. First, define the function $V : \mathbb{R}^2 \rightarrow \mathbb{R}_{\geq 0}$ as

$$V(x) = \int_0^{x_2} \widehat{G}(\xi) \xi d\xi + x^\top P x \quad (5)$$

$$\text{where } P = \frac{\gamma}{2} \begin{bmatrix} 1 & \epsilon \\ \epsilon & \epsilon \end{bmatrix} \succ 0, \epsilon \in (0, 1) \quad (6)$$

and define the functions $W_1, W_2, W_3 : \mathbb{R}^2 \rightarrow \mathbb{R}_{\geq 0}$ as

$$W_1(x) = \frac{1}{2} \widehat{G}(0)(x_2^+)^2 + x^\top P x \quad (7)$$

$$W_2(x) = \frac{1}{2} \widehat{G}(x_2)x_2^2 + x^\top P x \quad (8)$$

$$W_3(x) = \gamma(1 - \epsilon) \widehat{G}(x_2)x_2^2 + x^\top Q x \quad (9)$$

$$\text{where } Q = \epsilon\gamma^2 \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \succeq 0 \quad (10)$$

and define the equilibrium set E as

$$E = \{x_2 = -x_1 \text{ and } x_1 \geq 0\} \quad . \quad (11)$$

Lemma 1: The system defined by (4), with any choice of $\gamma, g \in \mathbb{R}_{>0}$ and $G : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ non-decreasing, has an equilibrium set E , and V as defined above satisfies

$$W_1(x) \leq V(x) \leq W_2(x) \quad (12)$$

$$\frac{d}{dt} V(x(t)) = -W_3(x(t)) \leq 0 \quad . \quad (13)$$

Additionally, W_1 and W_2 are positive definite functions and W_3 is a positive semi-definite function that is equal to zero only when $x \in E$.

Proof: The derivative of the first state component $\dot{x}_1 = \widehat{G}(x_2)x_2$ is equal to zero if and only if $x_2 \leq 0$, and the derivative of the second state component $\dot{x}_2 = -\gamma(x_1 + x_2)$ is equal to zero if and only if $x_2 = -x_1$. Therefore, \dot{x} is zero if and only if $x \in E$.

Since \widehat{G} is non-decreasing and $\widehat{G}(x_2) = 0$ when $x_2 < 0$, we can bound the first term of V as follows

$$\int_0^{x_2} \widehat{G}(0) \xi^+ d\xi \leq \int_0^{x_2} \widehat{G}(\xi) \xi d\xi \leq \int_0^{x_2} \widehat{G}(x_2) \xi d\xi$$

$$\frac{1}{2} \widehat{G}(0) (x_2^+)^2 \leq \int_0^{x_2} \widehat{G}(\xi) \xi d\xi \leq \frac{1}{2} \widehat{G}(x_2) x_2^2 \quad ,$$

which gives us in the inequality shown in (12) by adding $x^\top P x$ to each term in the inequality above. To show (13), we take the derivative of V along trajectories of the system

$$\frac{d}{dt} V(x(t)) = \dot{x}_2 \widehat{G}(x_2) x_2 + \dot{x}^\top P x + x^\top P \dot{x} \quad , \quad (14)$$

the first term of which can be represented as

$$\begin{aligned} \dot{x}_2 \widehat{G}(x_2) x_2 &= -\gamma(x_1 + x_2) \widehat{G}(x_2) x_2 \\ &= -x^\top S(x) x \end{aligned}$$

with $S(x) = \gamma \widehat{G}(x_2) \begin{bmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & 1 \end{bmatrix}$. We can now write the derivative of V as

$$\frac{d}{dt} V(x(t)) = -x^\top Q(x) x \quad (15)$$

$$\text{with } Q(x) = S(x) - P \widehat{A}(x) - \widehat{A}^\top(x) P \quad . \quad (16)$$

Our choice of $P \succ 0$ in (6) was constructed so $Q(x)$ is a positive semi-definite matrix for all values of x and $x^\top Q(x) x$ is equal to zero if and only if $x \in E$. Plugging our choice of P (6) into (16), we obtain

$$Q(x) = \begin{bmatrix} \gamma^2 \epsilon & \gamma^2 \epsilon \\ \gamma^2 \epsilon & \gamma^2 \epsilon + \gamma(1 - \epsilon) \widehat{G}(x_2) \end{bmatrix} \quad ,$$

which means that

$$\begin{aligned} \frac{d}{dt} V(x(t)) &= -\gamma(1 - \epsilon) \widehat{G}(x_2) x_2^2 - x^\top Q x \\ &= -W_3(x) \leq 0 \quad . \end{aligned}$$

Now that we have the Lyapunov-like function V and the comparison functions W_1, W_2, W_3 , we can prove the stability of the closed-loop dynamics.

Theorem 1 (Stability of Human Pointing with Acceleration): For the system defined by (4), with any choice of $\gamma, g \in \mathbb{R}_{>0}$ and $G : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ non-decreasing, the state globally asymptotically converges to the equilibrium set $E = \{x_2 = -x_1 \text{ and } x_1 \geq 0\}$.

Proof: From the existence of the function V , which is positive definite and radially unbounded, and \dot{V} is strictly less than zero except when $x \in E$ where $\dot{V} = 0$, we know that the system will globally converge to the equilibrium set E .

V. STATE AND POINTING PERFORMANCE BOUNDS

We now know that the closed loop pointing system will be stable with acceleration; however, we wish to get a better idea of how the system will perform. We will specifically look at two performance metrics of pointing systems: pointing accuracy and pointing speed.

From the definition of the closed-loop dynamics (4), the first state component x_1 is defined as the (perceived) pointer position and the second state component x_2 is an internal state that controls the switching of the dynamics between moving towards the target and stopping. We have already assumed $x_1(0) < 0$, i.e. the starting pointer position is behind the target position, and we will typically assume $x_2(0) = 0$, which means the pointer will start with velocity equal to zero. From this starting state, x_2 moves above zero and causes the pointer position x_1 to move towards the target, which is at the origin. Eventually, x_2 will begin decreasing and cross zero again, which corresponds to when the user perceives that the target has been reached, and the pointer position x_1 will stop moving. We refer to the time it takes the pointer to come to rest as the stopping time. After the stopping time, there may be some error between the target position and pointer position, and we call this error the overshoot.

We define the stopping time T and the overshoot \mathcal{O} as:

$$T = \inf_{t>0} t \quad \text{such that} \quad \lim_{\tau \rightarrow t} x_2(\tau) = 0 \quad (17)$$

$$\mathcal{O} = \lim_{\tau \rightarrow T} x_1(\tau) \quad (18)$$

Given a particular acceleration profile G , we could calculate the stopping time T and the overshoot \mathcal{O} by integrating the dynamics, but this method could be computationally expensive if we want to compare a large number of different acceleration profiles and other parameter choices; instead, we would like to find bounds on the performance that are simple to analyze and compute. We will derive analytical expressions time-dependent bounds on the state $x(t)$

$$\underline{x}(t) \leq x(t) \leq \bar{x}(t) \quad (19)$$

and bounds on the stopping time T and overshoot \mathcal{O}

$$\begin{aligned} \underline{T} &\leq T \leq \bar{T} \\ \underline{\mathcal{O}} &\leq \mathcal{O} \leq \bar{\mathcal{O}} \quad . \end{aligned} \quad (20)$$

We begin by using a similar method as we used in the proof of Lemma 1, which relies on the fact that the nonlinear term in the dynamics $\widehat{G}(x_2)$ can be bound when given \underline{k}, \bar{k} such that

$$\underline{k} \leq \widehat{G}(0) \quad \bar{k} \geq \sup_{t \geq 0} \widehat{G}(\bar{x}_2(t)) \quad (21)$$

then for any $x_2 > 0$

$$\widehat{G}(x_2(t)) \in [\underline{k}, \bar{k}] \quad . \quad (22)$$

Note that \bar{k} is guaranteed to exist, since given some initial state $x(0) = [-a, 0]^\top$, the state must remain in the bounded invariant set $V(x) \leq V(x(0))$, which is a direct result of Lemma 1. A very conservative choice produced using this method is $\bar{k} \geq \widehat{G}(a)$. Later we will show how to produce a less conservative choice of \bar{k} .

We can now derive time-dependent bounds $\bar{x}(t), \underline{x}(t)$ on the state $x(t)$ that are functions of \underline{k} and \bar{k} . First, define the

matrices

$$M_1 = \begin{bmatrix} 0 & \bar{k} & 0 & 0 \\ 0 & -\gamma & -\gamma & 0 \\ 0 & 0 & 0 & \underline{k} \\ -\gamma & 0 & 0 & -\gamma \end{bmatrix} \quad (23)$$

$$M_2 = \begin{bmatrix} 0 & \bar{k} & 0 & 0 \\ 0 & -\gamma & -\gamma & 0 \\ 0 & 0 & 0 & 0 \\ -\gamma & 0 & 0 & -\gamma \end{bmatrix} . \quad (24)$$

Theorem 2 (State and Performance Bounds): Given the initial state value $\underline{x}(0)$ with $x_1(0) < 0$ and $x_2(0) = 0$, and given bounds \bar{k} and \underline{k} as in (21), then state bounds $\phi(t) = [\bar{x}^\top(t), \underline{x}^\top(t)]^\top$ of the pointing system (4) can be chosen as

$$\phi(t) = \begin{cases} e^{M_1 t} \phi(0) & , t \in [0, \underline{T}] \\ e^{M_2(t-\underline{T})} e^{M_1 \underline{T}} \phi(0) & , t \in [\underline{T}, \bar{T}] \end{cases} , \quad (25)$$

where $\phi(0) = [x^\top(0), x^\top(0)]^\top$. The corresponding performance bounds satisfy

$$\begin{aligned} \underline{x}_2(\underline{T}) &= 0 & \underline{\mathcal{O}} &= \underline{x}_1(\underline{T}) \\ \bar{x}_2(\bar{T}) &= 0 & \bar{\mathcal{O}} &= \bar{x}_1(\bar{T}) \end{aligned} , \quad (26)$$

which also implies

$$\begin{aligned} \bar{T} &= \underline{T} + \frac{1}{\gamma} \ln \left[1 + \frac{\bar{x}_2(\underline{T})}{\underline{x}_1(\underline{T})} \right] \\ \bar{\mathcal{O}} &= \underline{x}_1(\underline{T}) + \frac{\bar{k}}{\gamma} [\bar{x}_2(\underline{T}) - \gamma(\bar{T} - \underline{T}) \underline{x}_1(\underline{T})] \end{aligned} . \quad (27)$$

Proof: The bounds $\phi(t)$ form a time-varying rectangular set which we will show is invariant because along each edge of the set, the rate of change of the bounds $\dot{\phi}(t)$ along the normal direction of the bounds exceeds the rate of change $\dot{x}(t)$ of any state that could feasibly be on that edge. Given $x_2(t) \geq 0$, then along each of the specified edges we have

$$\begin{aligned} \text{Right: } \dot{x}_1 &= \hat{G}(x_2) x_2 \leq \bar{k} \bar{x}_2 = \dot{\bar{x}}_1 \\ \text{Top: } \dot{x}_2 &= -\gamma(x_1 + \bar{x}_2) \leq -\gamma(\underline{x}_1 + \bar{x}_2) = \dot{\bar{x}}_2 \\ \text{Left: } \dot{x}_1 &= \hat{G}(x_2) x_2 \geq \underline{k} \underline{x}_2 = \dot{\underline{x}}_1 \\ \text{Bottom: } \dot{x}_2 &= -\gamma(x_1 + \underline{x}_2) \geq -\gamma(\bar{x}_1 + \underline{x}_2) = \dot{\underline{x}}_2 \end{aligned} .$$

All of these also hold for $x_2(t) < 0$, except along the left edge $\dot{x}_1 = \dot{\underline{x}}_1 = 0$. Therefore, if the state $x(t)$ starts within the bounds $\phi(t)$, then the definition of $\dot{\phi}(t)$ above will ensure that $x(t)$ stays within $\phi(t)$ for all t . Combining these relationships

$$\dot{\phi}(t) = \begin{cases} M_1 \phi(t) & , \text{ if } \underline{x}_2(t) \geq 0 \\ M_2 \phi(t) & , \text{ if } \underline{x}_2(t) < 0 \end{cases} , \quad (28)$$

which produces (25) when integrated.

The performance bounds shown in (26) can be shown simply by the definitions of \underline{T} and $\underline{\mathcal{O}}$. The simpler forms of \bar{T} and $\bar{\mathcal{O}}$ in (27) were derived by directly integrating the terms in $\dot{\phi} = M_2 \phi(t)$, which is relatively straightforward since the terms are mostly decoupled. ■

It would be useful to derive simplified expressions for \underline{T} and $\underline{\mathcal{O}}$ that are similar to (27) by analytically solving $\dot{\phi} =$

$M_1 \phi(t)$. However, we could not find any simple closed form way to express \underline{T} and $\underline{\mathcal{O}}$, due to the added complexity caused by the extra coupling term in M_1 . We can still easily compute \underline{T} and $\underline{\mathcal{O}}$ given the previous theorem, and we can potentially analyze how \underline{T} and $\underline{\mathcal{O}}$ relate to other parameters, but we have not yet found analytic expressions for these terms.

Now we have time-dependent bounds on the state and performance that depend on \bar{k} , a bound on the nonlinear term in the dynamics. We previously mentioned how we could pick \bar{k} based on our Lyapunov function $V(x)$, but we can also pick \bar{k} based on the time-dependent bounds on the state.

Lemma 2: The bounds $\bar{x}_2(t)$ as in Theorem 2 obtain a maximum value when $t = T^*$ where

$$[0 \quad 1 \quad -1 \quad 0] e^{M_1 T^*} \phi(0) = 0 . \quad (29)$$

Writing $\bar{x}_2^* = \bar{x}_2(T^*)$ as a function of \bar{k} , and given some \bar{k}_0 that satisfies (22), if \bar{k} satisfies

$$\bar{k} \geq \hat{G}(\bar{x}_2^*(\bar{k}_0)) , \quad (30)$$

then \bar{k} also satisfies (22).

Proof: $\bar{x}_2(t)$ achieves its maximum when $\dot{\bar{x}}_2(t) = -\gamma(\bar{x}_2(t) + \underline{x}_1(t)) = 0$, so $\bar{x}_2(t) + \underline{x}_1(t) = 0$ or $[0 \quad 1 \quad -1 \quad 0] \phi(t) = 0$.

From Theorem 2, $x_2(t) \leq \bar{x}_2(t, \bar{k}_1) \leq \bar{x}_2^*(\bar{k}_1)$, where we slightly abuse notation here by writing these bounds as functions of \bar{k} . Since \hat{G} is increasing, $\hat{G}(x_2(t)) \leq \hat{G}(\bar{x}_2^*(\bar{k}_1))$, therefore any \bar{k} such that $\bar{k} \geq \hat{G}(\bar{x}_2^*(\bar{k}_1))$ satisfies (22). ■

This lemma provides us with a method to iteratively reduce the value of \bar{k} , which will produce tighter bounds on the state and performance. We have the same difficulty with the expression for \bar{x}_2^* as we did with the expressions for \underline{T} and $\underline{\mathcal{O}}$ in Theorem 2; a closed form expression is not easily obtainable. But again, given Lemma 2 it is fairly simple to compute the value of \bar{x}_2^* , and then compute \bar{k} , and the relationship shown in this lemma may still have some other analytical value.

The implications of Theorem 2 are, given an acceleration profile, we can directly compute the performance bounds for the closed-loop pointer acceleration system, without needing to integrate any trajectories. The only values that we must compute to obtain the performance bounds are evaluating some matrix exponentials, which can be computed efficiently using Jordan decomposition, and possibly calculating \bar{k} , which could involve an iterative minimization process using Lemma 2. While the dependence of these performance bounds on the acceleration profile G is not trivial, it is simple enough that there may potentially be a way to determine the dependence analytically, which would open up many different possibilities for designing G to optimize some metric using the performance bounds.

VI. SIMULATION

We use simulation to further examine how pointer acceleration affects the performance of pointing systems, as indicated by results derived previously. Figure 2 shows a simulated pointer trajectory for a given acceleration profile

along with the time-dependent bounds on the trajectories. The acceleration profile used was linear with $k_1 = 1$ and $k_2 = 0.1$, and with the system parameters $\gamma = 4$ and $g = 2$. These results show that the trajectories do stay within the time-dependent trajectory bounds derived in this paper.

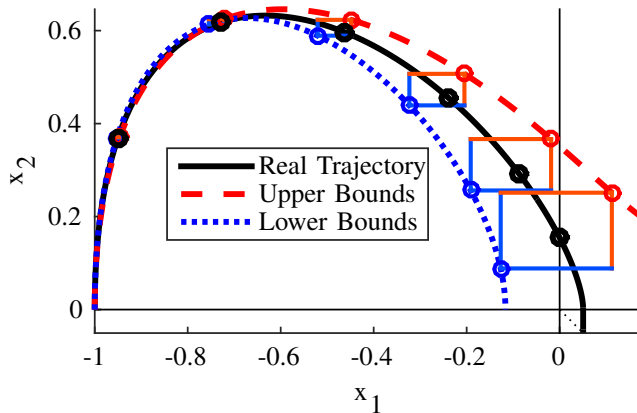


Fig. 2: Simulated trajectory of closed loop pointing system with acceleration (solid black line) along with time-dependent upper state bounds (dashed red) and lower state bounds (dotted blue) on the trajectory. The state bounds are also plotted at a few points in time as bounding boxes with the state at the corresponding time indicated by black dots.

VII. CONCLUSIONS AND FUTURE WORK

In this paper we presented a new model for pointer acceleration and showed some of its basic properties when connected in closed loop to the VITE model for human pointing. Specifically, we showed that the closed loop system was stable and provided some bounds on the state and performance.

One interesting direction for this work is to apply these results to the design of pointing acceleration implementations; particularly, these results may provide some insight into how to choose acceleration profiles to better suit performance goals and human operator properties. These results could also potentially be extended in a number of different directions including modifying the pointing acceleration model to include other commonly applied pointer effects, such as input

smoothing and pointer prediction. Future work could also incorporate a more complex model of human pointing behavior to capture other effects, such as corrective motions and workspace constraints. Analysis of the case with feedback delays and perturbations is also important, as it has direct relevance to practical applications such as teleoperation. Additional work on the experimental side of this research could also be valuable to validate the conclusions of this work and to expose other potential research directions.

REFERENCES

- [1] (2002, Oct.) Pointer Ballistics for Windows XP. [Online]. Available: <http://web.archive.org/web/20070523112807/http://www.microsoft.com/whdc/device/input/pointer-bal.msp>
- [2] (2008, May.) Mouse Acceleration Preference Pane for Mac OS X. [Online]. Available: <http://www.triq.net/articles/mouse-acceleration-preference-pane-mac-os-x>
- [3] (2013, Jul.) X.Org PointerAcceleration Development Documentation. [Online]. Available: <http://xorg.freedesktop.org/wiki/Development/Documentation/PointerAcceleration/>
- [4] R. Shadmehr and S. P. Wise, *The computational neurobiology of reaching and pointing: a foundation for motor learning*. MIT Press, 2005.
- [5] P. M. Fitts, "The information capacity of the human motor system in controlling the amplitude of movement." *Journal of experimental psychology*, vol. 47, no. 6, pp. 381–391, 1954.
- [6] D. Bullock and S. Grossberg, "Neural dynamics of planned arm movements: emergent invariants and speed-accuracy properties during trajectory formation." *Psychological review*, vol. 95, no. 1, pp. 49–90, Jan. 1988.
- [7] D. Beamish, C. Peskun, and J. Wu, "Critical delay for overshooting in planned arm movements with delayed feedback," vol. 48, pp. 22–48, 2005.
- [8] D. Beamish, S. A. Bhatti, I. S. MacKenzie, and J. Wu, "Fifty years later: A neurodynamic explanation of Fitts' law." *Journal of the Royal Society, Interface / the Royal Society*, vol. 3, no. 10, pp. 649–54, Oct. 2006.
- [9] D. Beamish, S. Bhatti, J. Wu, and Z. Jing, "Performance limitations from delay in human and mechanical motor control." *Biological cybernetics*, vol. 99, no. 1, pp. 43–61, July 2008.
- [10] D. Beamish, S. Bhatti, C. S. Chubbs, I. S. MacKenzie, J. Wu, and Z. Jing, "Estimation of psychomotor delay from the Fitts' law coefficients." *Biological cybernetics*, vol. 101, no. 4, pp. 279–96, Oct. 2009.
- [11] P. Varnell and F. Zhang, "Dissipativity-Based Teleoperation with Time-Varying Communication Delays," no. 2007, 2013.
- [12] (2006, Nov.) How to Disable Mouse Acceleration. [Online]. Available: <https://support.microsoft.com/en-us/kb/149228>
- [13] mouse_event function. Windows Dev Center. [Online]. Available: <https://msdn.microsoft.com/en-us/library/windows/desktop/ms646260.aspx>