Dynamics of Pointing with Pointer Acceleration

Anonymous

Abstract. In this paper we investigate the dynamics (including velocities and accelerations) of mouse pointing when Pointer Acceleration (PA) functions are used. We also propose a simple model for these dynamics from a control theoretic perspective. The model allows us to simulate the effect of PA functions on pointing dynamics. In particular, it reproduces and explains many important phenomena we observe in pointing dynamics with PA functions. These include: (1) Pointer position, velocity, and acceleration over time, (2) Different accelerations when moving in different directions, and the resulting mouse drift when using PA functions, (3) Discontinuous jumps in phase space and associated acceleration peaks in Hooke plots when using Step PA functions. We identify parameters of the model using a reciprocal pointing task with the mouse controlling a pointer on a computer screen using sigmoid and step PA functions and constant gain. Our model explains the human-computer system including the PA function as a closed-loop dynamical system. In particular, we use a second-order model resembling a spring-mass-damper system (second order lag). Our model explains and allows to simulate the role of PA functions in pointing, including the phenomena described above.

1 Introduction

The contribution of this paper is an investigation of pointing dynamics using Pointer Acceleration (PA) functions. PA functions adjust the mouse gain depending on mouse velocity. PA functions are implemented in all major operating systems and are used by millions of users on a daily basis. Given their ubiquity and importance, it is very surprising how little we know about them. Consider the simple case of pointing with the Trackpoint under Windows. The Trackpoint microcontroller applies a force-to-motion function to the applied force that has been optimized for pointing at text targets [20]. Windows treats the Trackpoint as a mouse and applies to the output of this function a PA function that has presumably been optimized for mouse movements. We do not know how these transfer functions interact to produce the final pointer movement. Similarly, Windows ignores the resolution of mice when applying PA functions, forcing users to reduce the resolution on the mouse microcontroller to obtain usable mouse gains. We can reverse engineer the PA functions implemented in major operating systems [5], but do not know anything about their design rationale. Transfer functions are crucial not only for Desktop settings, but also for mid-air gestures and Virtual Reality interaction [19,10,14,11].

In addition to analyzing pointing dynamics, we also provide a simple model of pointing dynamics with PA functions. While our model is certainly not at the point where it can actually be used for the optimization of transfer functions, it aims to be a first step in that direction. At the current point, the model can quantitatively predict pointer

position and velocity over time with some accuracy. More importantly, it reproduces and explains many important phenomena we observe in pointing dynamics with PA functions qualitatively. These include: (1) Shape of pointer position, velocity, and acceleration over time and space (2) Different accelerations when moving in different directions, and the resulting mouse drift when using PA functions (3) Discontinuous jumps in phase space and associated acceleration peaks in Hooke plots when using Step PA functions

2 Related Work

2.1 Traditional Understanding of Pointing in HCI

Pointing is often understood as a discrete event in HCI. A pointer is moved from a start position to a target of a certain width W at a distance D. When the target is reached, a delimiter, such as a mouse click, terminates the movement. Of interest is usually only the overall movement time MT and error rate. The dynamics of the movement itself, such as the position, velocity, and acceleration over time, of both pointer and pointing device, are often not investigated. The model for movement time most often used in HCI is Fitts' law [8,9]. It allows to predict movement time as $MT = a + b\log_2(\frac{D}{W} + 1)$ in the Shannon formulation [16]. The constants a and b are specific for a certain user and input device, but independent from distance to and width of the target. They can be obtained from pointing data of a user by linear regression. Fitts' derived the relationship from the Shannon-Hartley theorem [22]. This theorem gives the channel capacity of a continuous channel with Gaussian noise. Conclusions from the channel capacity to the actual movement that takes place are not straightforward. Crossman and Goodeve [7] proposed an alternative derivation of Fitts' law based on assumptions about the actual pointer movement. They assume that pointing consists of a series of sub-movements of equal duration and error, until the target is reached. Crossman and Goodeve do not make further predictions about the dynamics of pointing. Meyer [17] developed an extension of the Crossman Goodeve model that assumes that humans optimize the durations of sub-movements to minimize the overall movement time under noise.

2.2 Current understanding of Transfer Functions in HCI

Transfer functions map data from an input device (e.g., mouse) to the interface (e.g., pointer movement). As such they are an essential component of systems that enable interaction of humans with computers. Design of transfer functions has a long history in HCI [20,1,10,11,14,19,18]. The common design process of such functions is to select a class of functions that can be parameterized with a small number of parameters. The parameter space is then commonly searched manually by trying various parameter settings with the TF designers themselves, their colleagues, or experiment participants. One important case of transfer functions are pointer acceleration (PA) functions. PA functions increase the mouse gain with increasing mouse velocity. Jellinek and Card [13] performed three experiments investigating PA functions. The tested PA functions did not result in different performance. They concluded that PA functions do not improve performance, and would "violate Fitts' law" if they would. They also concluded that

PA functions are preferred by some users because they "lower the device footprint" and require less clutching (repositioning of the mouse).

Nancel et al. [18] analyze transfer functions for mid-air pointing on display walls. They present a theoretical analysis of dual-precision pointing techniques based on Fitts' law. Further, they present detailed tuning guidelines for the parametrization of sigmoid pointer acceleration functions.

The most thorough investigations of transfer functions in HCI are by Casiez et al. In [6], they compare movement times in reciprocal pointing for static gains (1,2,4,6,8,12) and six levels for the PA function implemented in Windows XP/Vista. They find that PA results in 3.3% faster pointing, and 5.6% for small targets. They also propose a theoretical explanation based on Meyer et al.'s stochastically optimized sub-movement model. They denote by CD_D the average gain during the ballistic phase (surge), and by CD_W the average gain during the corrective phase. They deduce that the effective ID in motor space is $ID_{mot} \approx ID + log_2 \frac{CD_W}{CD_D}$. This formula gives an indication of why PA functions should work. However, it suffers from two fundamental problems. First, the effective ID ID_{mot} can be reduced arbitrarily by choosing very large CD_D and very small CD_W. This is not consistent with observations. Casiez et al. argue that the observed PA functions provide lower benefits than predicted "possibly because PA results in increased target overshooting".

We argue it is precisely this overshooting which provides the fundamental performance limitations of PA functions. This overshooting is a dynamic behavior which can be modeled by control theoretical models, but not by Fitts' law analysis.

2.3 Dynamical Systems Perspective

In contrast to modeling pointing as a discrete movement or series of sub-movements, it has also been modeled as a dynamical system. Dynamical systems can be modeled as differential equations, which have as a solution a prediction of pointer position over time [23,**?**]. For example, Jagacinsky and Flach [12] show how human pointing can be modeled as a simple first order lag or second order lag (2OL). They also show how Fitts' law can be derived from either of these models.

The VITE model [4] is a neural network model of pointing. The model is governed by the differential equations $\frac{dV}{dt} = \alpha(-V + T - P)$ and $\frac{dP}{dt} = G[V]^+$. $T(t)$ is the target position, $V(t)$ "the activity of the agonist's DV population", or in other words, related to the velocity of the end-effector, $P(t)$ is "the activity of the agonist's PPC population", or in other words, related to the position of the end-effector. $G(t)$ is the "GO signal", which can be used to modulate the mapping from V to P. The model can be used to predict end-effector position, velocity, and acceleration over time given a start and target position.

Beamish et al. [2] have extended the model with a delay term and have proven that Fitts' law follows from this model.

Very recently, the model has been extended to include Pointer Acceleration functions [24]. It was shown that the resulting model is stable under any PA function, and theoretical performance bounds of the model have been derived. The main difference of that work to our paper is that Varnell and Zhang's work analyzes the theoretical properties of their model. While proving stability and performance bounds of models is a core task in

control theory, it is less directly applicable to HCI. In practice we don't observe unstable human-computer system behavior (e.g., wild oscillations of mouse users). In contrast, our work is of more empirical nature. We present a dataset of actual user behavior, a model of pointing with PA functions that can replicate phenomena we observe in that dataset, a system identification process that can learn parameters of the model from the dataset, and an analysis of model behavior in comparison to user behavior.

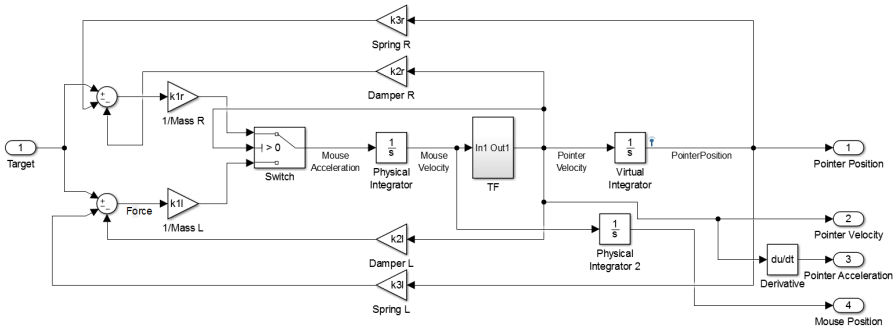

Fig. 1. Our control theoretical model of pointing with a Pointer Acceleration function. The model predicts pointer position, velocity and acceleration as well as mouse position over time given a target position. The model can loosely be interpreted as switching between two spring-mass-damper systems depending on the movement direction of the pointer. Further, the velocity of the system gets modified by a nonlinearity within the loop, the Pointer Acceleration function, breaking the strict spring-mass-damper interpretation.

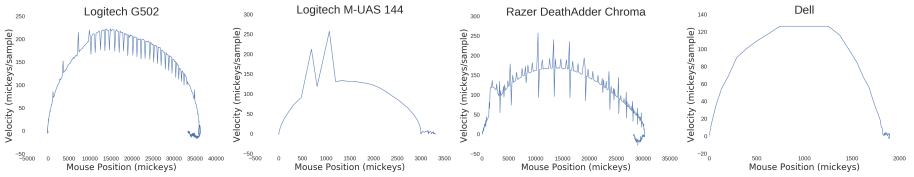

Fig. 2. Noise for different mice. Logitech G502, Logitech UAS144, Razer DeathAdder, and a Dell mouse, respectively. The noise gets amplified by PA functions. In case of Step PA functions, the gain might even oscillate at noise frequency if mouse velocity is close to the threshold.

3 Mouse Noise and PA Functions

The analysis of pointing dynamics and transfer functions is complicated by sensor noise. Mouse data is very noisy. Furthermore, different mouse models exhibit widely different noise characteristics. Raw mouse data (phase space plot: velocity plotted over position)

for four commercial mice is shown in Figure 2. While the Logitech G502 exhibits a relatively clean and high-resolution velocity profile, velocity spikes appear at changing regular rates. The Razer DeathAdder has a similarly high resolution sensor, but exhibits much more high frequency noise. The velocity bump at the beginning of the movement remains inexplicable. More conventional mice like the Logitech UAS144 have much lower resolution. Also, the mouse seems to loose tracking accuracy at high velocities. For everyday pointing, this noise is usually not noticeable, because the integration of pointer velocity to position smoothes noise out. However, for the analysis of pointing dynamics, this noise is critical. Furthermore, the noise gets increased considerably when using PA functions, because velocity spikes get multiplied with a higher gain factor. When using Step PA functions, the gain can even oscillate between low and high gain with the noise frequency. To address this issue, in this paper we apply a median filter and a second order IIR low-pass filter to the mouse velocity *before* applying the PA function. In practice, this would not be done because it makes the selection of individual pixels more difficult. For the analysis of pointing dynamics, especially using PA functions, this step is however necessary to be able to see the patterns in the noise.

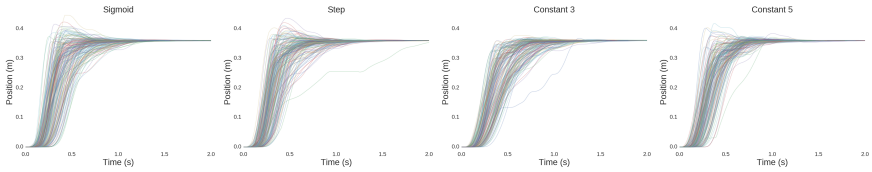

Fig. 3. Overview of time series of pointer position for all participants using the different transfer functions.

4 Dataset

We captured data using the *serial pointing task* [8], where a user clicks alternatingly at two one-dimensional targets. To improve replicability, we will release the dataset and model of this paper when it is published.

4.1 Method

Participants 12 unpaid participants (3 female, mean age 30.6 years (SD 8.1), all normal or corrected to normal eyesight, all expert computer users) participated in the study. They had on average 16.4 years (SD 6.5) of experience of mouse use, and currently used a mouse on average for 37.6 hours (SD 31.6) per week. While one participant was left-handed, all preferred to use the mouse with the right hand and used the right hand in the experiment.

Task and Materials Two one-dimensional targets were displayed. The task was to click on the targets serially. A new trial started as soon as the user clicked on the previous target. Missed trials were not repeated, but annotated in the dataset.

The *condition* of the experiment is pointer acceleration function. The pointer acceleration function is varied between blocks, but kept constant within each block. We cover 4 different pointer acceleration functions. The *sigmoid* function provides a gain of 3 when the mouse velocity is below 10 cm/s, and a gain of 5 when the mouse velocity is above 30 cm/s. Between these points, the gain increases linearly. The *step* function switches instantly between gains 3 and 5 at 20 cm/s. In addition, *constant gain* of *3* and *5* was tested. These TFs were selected for their simplicity and because they are the classical examples of PA functions. We did not test PA functions implemented in current operating systems because such results might be outdated soon. We chose the parameters to be consistent between the different functions and be usable for pointing in an iterative tuning process. Each condition is repeated for 50 trials. The order of conditions was counterbalanced using a Latin square.

Procedure Users were asked to adjust table, display and mouse pad to their preferences. All users were resting their palm on the mouse pad. Participants were introduced to the task and completed a training phase for all conditions before starting the experiment. Users trained for 10 trials in each condition. Users took a break after each condition. They were asked to stretch their limbs and relax briefly. Users were asked to adjust mouse position in the first trial of each condition and to avoid clutching during the experiment. To enable this, we used a very large mouse pad (900x450mm). Users were asked to click on the targets as quickly as possible while maintaining an error rate of below 5%. Targets were shown at a distance of 35.98 cm (1300 px) with a target width of 1.38 mm (5 px), leading to an index of difficulty (ID) of $log_2(\frac{D}{W}+1) = 8.028$.

Apparatus Data was captured on a Dell Precision 7810 PC with an AOC G2460PQU monitor (24", 1920x1080 px resolution, 140 Hz, no Vsync). The Logitech G502 mouse was used with no additional weights. The software used `libpointing.org` [5] for accessing the raw mouse data, instead of using the mouse data that is provided preprocessed by the operating system. The raw mouse data was filtered with a median filter with window size 3 to filter outlier samples (visible for Logitech G502 in Figure 2). The resulting signal was filtered with a low-pass IIR filter with a critical frequency of 15 Hz and a quality factor of 1. Meaningful frequencies in human movement are below this frequency [21], so that they were preserved in the signal. Filtering was performed to remove high-frequency noise components in the mouse, which would otherwise hinder the investigation of the impact of pointer acceleration functions on pointing dynamics. OpenGL (glfw3) was used for low-latency graphics generation. The program was running at approximately 2000 Hz, such that the frequency of the overall apparatus was limited by the 140 Hz of the monitor. Mouse events were logged as they were delivered to the program by libpointing. Data was captured under Microsoft Windows 10. Latency was measured with a Sony DSC-RX10M2 camera at 1000 FPS. The mouse was hit with a hard object at high speeds, and the number of frames from impact to pointer motion

on the display was counted. The latency was 25 ms. The *x*-dimension of the mouse was used to move a white crosshair pointer (1px wide) that only moved horizontally.

4.2 Preprocessing

In order to facilitate further analysis, data was preprocessed. We dropped the first 20 trials from each condition. This was done to remove trials where substantial learning of the new PA function was taking place. Naive calculation of derivatives (pointer acceleration) from mouse movement data would greatly increase any remaining noise in the signal. Therefore, we filter the pointer velocity using a Savitzky-Golay filter with a 4th degree polynomial and a window size of 21 samples (20ms) to calculate pointer acceleration as the 1st derivative of pointer velocity. We used the same filter to calculate mouse acceleration from mouse velocity as filtered in the apparatus.

5 Summary Statistics

Figures 3, 4, and 5 provide an overview of the dataset. Figure 3 shows the pointer position over time for all participants and all trials separated by PA function. In this visualization it is very difficult to detect differences between the functions. The only visible difference is that with constant gain 3 there seems to be less overshooting of the target. Figure 4 shows the pointer velocity plotted over pointer position (left column) as well as mouse velocity plotted over mouse position (right column). Because position and velocity determine the state of an inertial (second order) system, this visualization is called a phase space plot for such systems. The differences between the PA functions become much more obvious in this visualization. In particular, the phase space plot for the Step PA function is very different from the others. Whenever the velocity crosses the step, there is a sudden jump in pointer velocity.

The center parts of the individual curves seem simply to be “lifted” up from their normal elliptical shape. Figure 5 shows the pointer acceleration plotted over pointer position (left column) as well as mouse acceleration over mouse position (right column). This type of visualization is called a Hooke plot. The individual acceleration and deceleration spikes when using the Step PA function are clearly visible. Although the acceleration when crossing the step happens instantaneously, it is smoothed in this plot. This is because the acceleration is calculated from a polynomial approximation of velocity to manage noise. Also the Sigmoid PA function creates acceleration and deceleration spikes (although smoother) that can be seen when following individual trajectories. The isolated spike in the plot for constant gain 5 is the only clutching event in the data.

Figure 6 shows the maximum pointer velocities achieved in the experiment, separated by PA function and whether the movement is to the left or to the right. Figure 6 shows the maximum pointer accelerations achieved in the experiment, separated by PA function and whether the movement is to the left or to the right. It can be seen that the PA functions achieve higher accelerations than constant gain. Maximum acceleration of the Step PA function is infinite and therefore not shown. The surge movement denotes the initial (often ballistic) movement towards the target. We calculate the surge movement

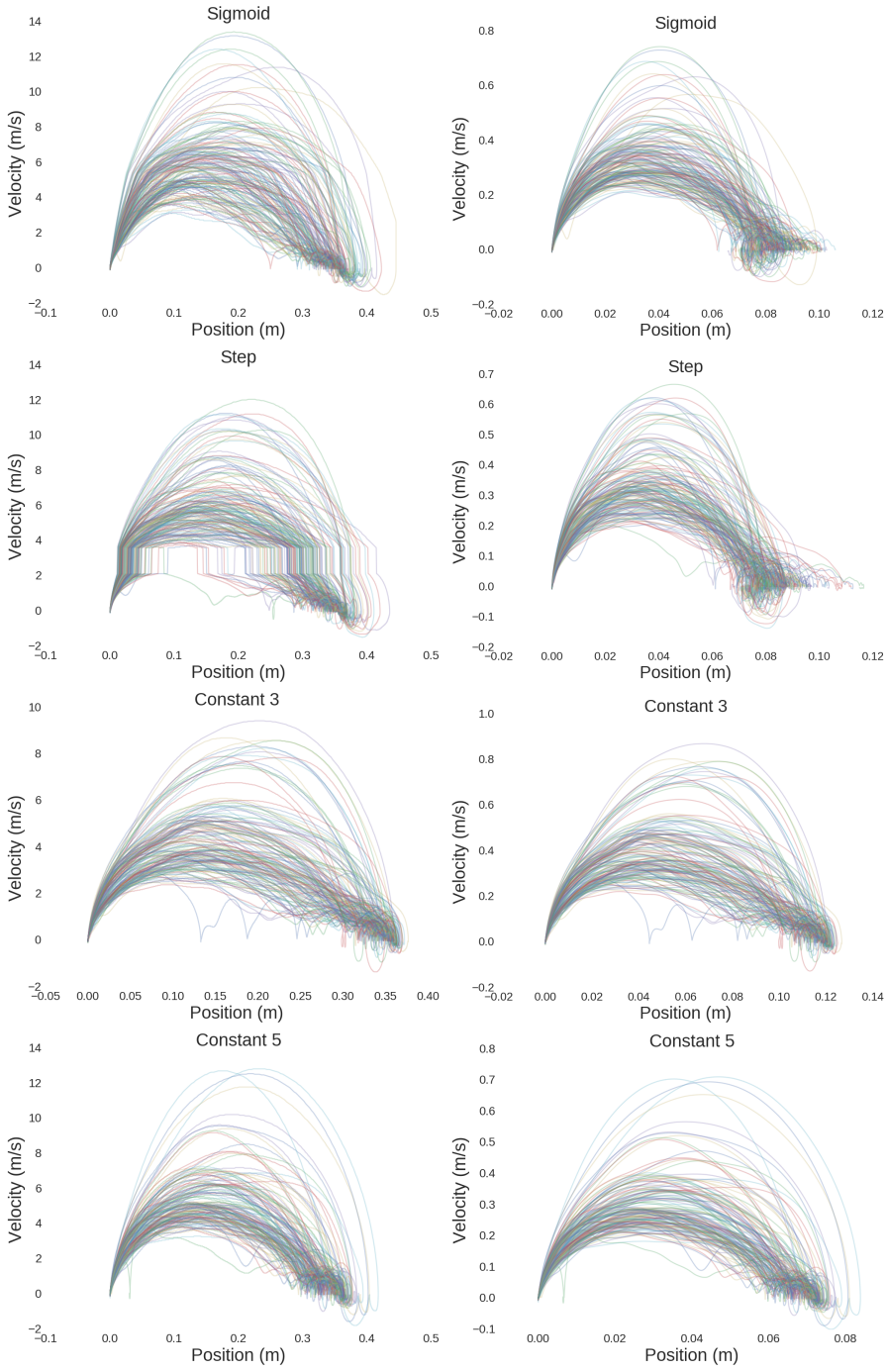

Fig. 4. Overview of phase space plots for all participants using the different transfer functions. Left column shows pointer velocity over pointer position, right column shows corresponding mouse velocity over mouse position. Note how the pointer profiles for the Step TF are “lifted up”. Also, the endpoints of mouse profiles with PA functions have more variance.

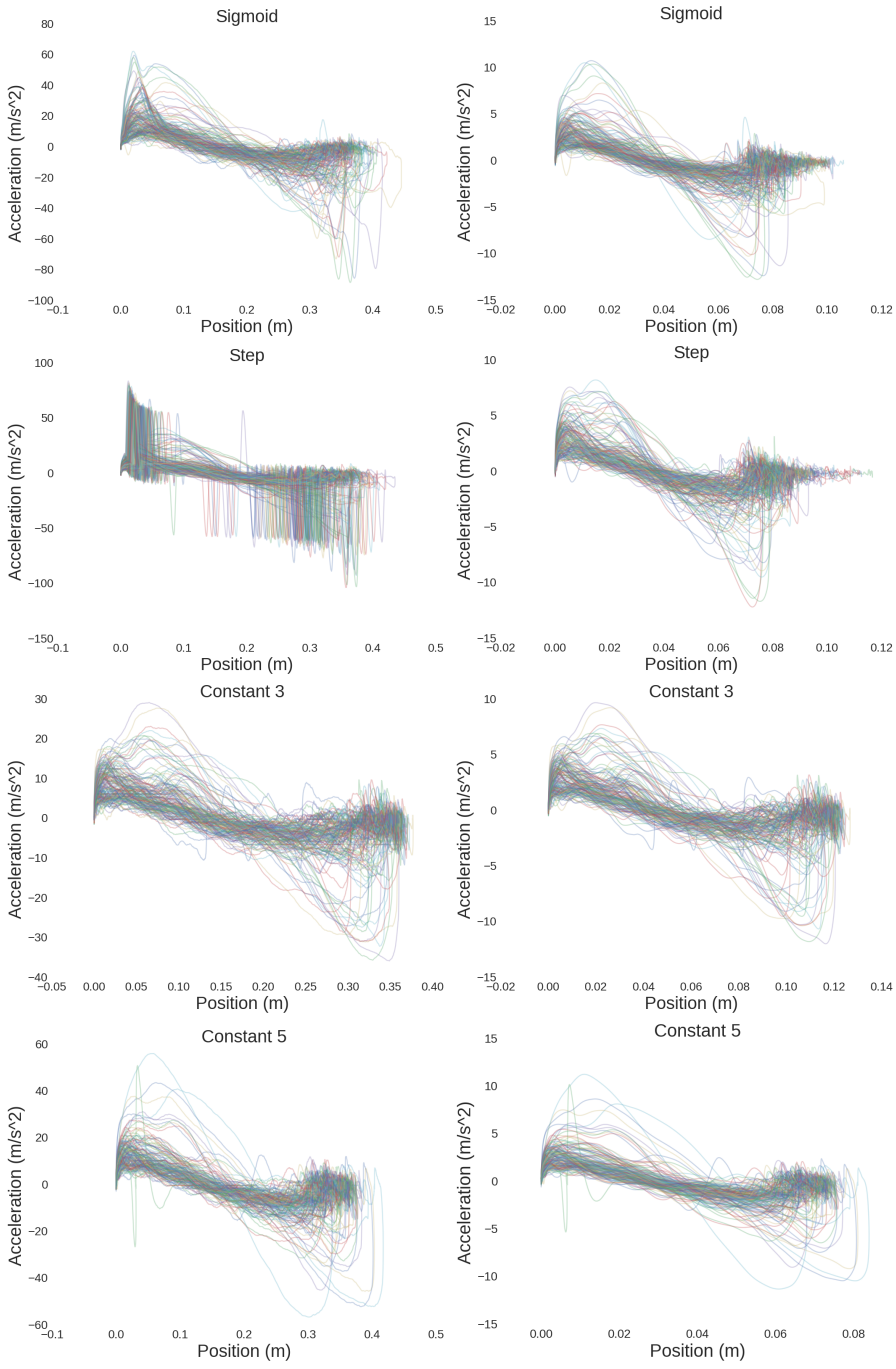

Fig. 5. Overview of Hooke plots for all participants using the different transfer functions. Left column shows pointer acceleration over pointer position, right column shows corresponding mouse acceleration over mouse position. The Step PA function creates infinite acceleration when the velocity crosses the Step (peaks smoothed here due to filtering). These peaks are still visible in the Sigmoid TF, but smoother.

X

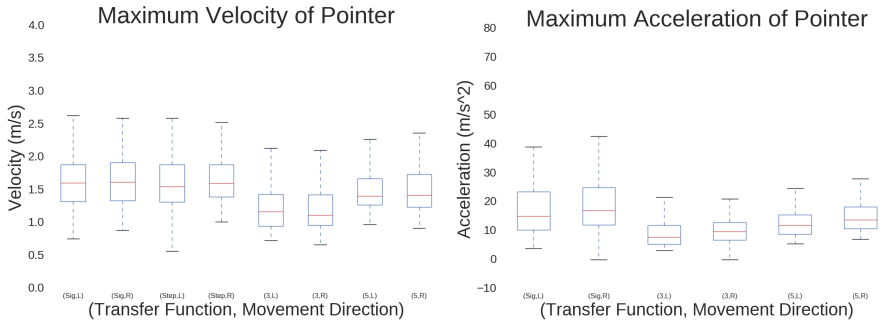

Fig. 6. Left: Maximum velocity of pointer separated by TF and movement direction. The PA functions reach higher velocities compared to constant gain. **Right:** Maximum pointer accelerations separated by TF and movement direction. Pointing with the Sigmoid PA function reaches higher accelerations than static gain. The Step PA functions creates infinite acceleration when crossing the step and is therefore not shown.

from mouse acceleration as follows. The point where the deceleration is larger than $-0.5m/s^2$ is detected. From there, the next point with zero acceleration is found, which we use as the end of the surge movement. Figure 7 shows the surge endpoints as a fraction of the overall distance to the target. Figure 7 shows the overshoot of targets by TF and movement direction. It can be seen that the target is often overshot by only a few millimeters.

We performed a number of statistical tests comparing characteristics of the individual trials by TF. A Friedman test revealed a significant effect of TF on maximum velocity ($\chi^2(3)$=404.04, $p < 0.01$). A post-hoc test using Wilcoxon tests with Bonferroni correction showed the significant differences between all pairs of TFs except Step PA and Sigmoid PA ($p<0.01$). A Friedman test revealed a significant effect of TF on maximum acceleration ($\chi^2(3)$=760.21, $p < 0.01$). A post-hoc test using Wilcoxon tests with Bonferroni correction showed the significant differences between all pairs of TFs. A Friedman test revealed a significant effect of TF on mean velocity ($\chi^2(3)$=760.21, $p < 0.01$). A post-hoc test using Wilcoxon tests with Bonferroni correction showed the significant differences between all pairs of TFs except Step PA and Sigmoid PA, and constant gain 3 and Step PA ($p<0.01$). A Friedman test revealed a significant effect of TF on overshoot ($\chi^2(3)$=17.998, $p < 0.01$). A post-hoc test using Wilcoxon tests with Bonferroni correction showed the significant differences between constant gain 3 and 5 ($p<0.01$). A Friedman test revealed a significant effect of TF on surge endpoint ($\chi^2(3)$=44.387, $p < 0.01$). A post-hoc test using Wilcoxon tests with Bonferroni correction showed the significant differences between all pairs of TFs except Step PA and Sigmoid PA and Sigmoid PA and constant gain 5 ($p<0.05$). We did not find an effect of PA function on trial time (Friedman, $\chi^2(3)$=6.137, $p > 0.05$).

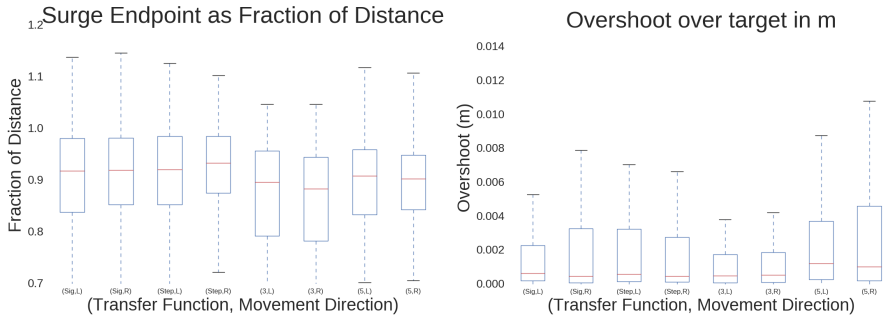

Fig. 7. Left: The surge denotes the initial ballistic pointer movement with a N-shaped acceleration profile at the end of which the acceleration drops to 0. The plot shows the endpoints of this ballistic movement as a fraction of distance to the target. **Right:** Overshoot denotes the amount by which the target is overshot at the maximum.

6 Control Theoretical Model of Transfer Functions

6.1 Modelling Process

We developed the model in an iterative process. In each iteration, the experimental design and apparatus were updated, data collected, data analyzed, the model designed, the model simulated, and model behavior analyzed. We iterated individual phases as well as the whole cycle. Key aspects of iteration were: selection of conditions and parameterization of PA functions, filtering of mouse data to detect phenomena caused by PA functions, detection and presentation of phenomena, and model design. The key tradeoff in the model design is between predictive accuracy and model complexity. For this paper, we decided to present the simplest possible model that can replicate the phenomena we are interested in to a satisfactory degree.

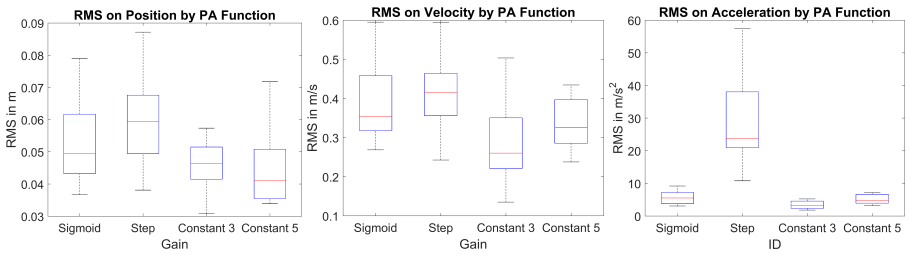

Fig. 8. RMSE of position (top left), velocity (top right) and acceleration (bottom) between model and user behavior for the Sigmoid PA function, Step PA function, constant gain 3 and constant gain 5.

6.2 The Model

Our model of pointing dynamics with PA functions (Figure 1) is a switching controller, switching between two second order lag (2OL) controllers based on movement direction. Additionally, between the two integrators (at the mapping from mouse velocity to pointer velocity), a non-linear pointer acceleration function is introduced. This PA function changes the gain based on mouse velocity. The PA function is implemented as a lookup table, which linearly interpolates between table points.

The model can be interpreted from a physical perspective. In this perspective, the pointer is interpreted as consisting of a mass m, a spring k_3, and a damper k_2. The spring can be thought of being attached to the target (in the user's mind) and the pointer. This spring then exerts a force on the user's arm that is proportional to the distance between pointer and target (consistent with Hooke's law). The motion of the pointer is damped, similar to friction which exerts a force on the arm that is proportional to the velocity of the pointer and opposes the direction of movement. These forces are added and translate into acceleration of the arm according to Newton's second law $F = ma$. The parameter $1/m$ is called k_1 in the model. The arm/mouse acceleration is then integrated to yield mouse velocity. Mouse velocity is then fed into the transfer function. The transfer function translates mouse velocity to pointer velocity by applying a speed dependent gain. This pointer velocity then gets multiplied with the damping parameter k_2 and exerts a proportional negative force on the arm. The pointer velocity also gets integrated to yield pointer position. The pointer position gets multiplied with the spring parameter k_3 and exerts a proportional force in the direction of the error. In summary, model behavior for moving in one direction can be given by the differential equation $k_1\ddot{y} = (target - k_3 y) - k_2\dot{y}$, where y is the pointer position. The parameters k_1, k_2 and k_3 are different depending on movement direction. The model discretely switches between the parameter sets depending on the velocity being above 0 (movement to the right) or below (movement to the left).

The model directly corresponds to the equilibrium point hypothesis of motor control [21]. In this hypothesis, it is assumed that the brain only sets the target position ("equilibrium point") of the system. The feedback loops and parameters are thought of describing the mechanical properties of the biomechanical system and reflex loops. Alternatively, it can be thought that the brain computes the error and controls the force exerted on the muscles directly. In any case, the model describes the overall system of brain, biomechanical system, reflex loops and computer, and the boundaries between these modules can be drawn in the model in many different ways.

6.3 Model Implementation

We implement the model in Simulink as a block diagram. Simulink is a Matlab extension that allows simulation of linear and non-linear systems. Because the transfer function introduces a nonlinearity inside the feedback loop, simulation of the system is not as straightforward as with linear systems, such as a pure 2OL. The discontinuous nature of the switching control and the transfer function influence the choice of solver for the model. We chose a fixed-step solver (ode5, Dormand-Prince) with a step width of 1 ms to simulate the model.

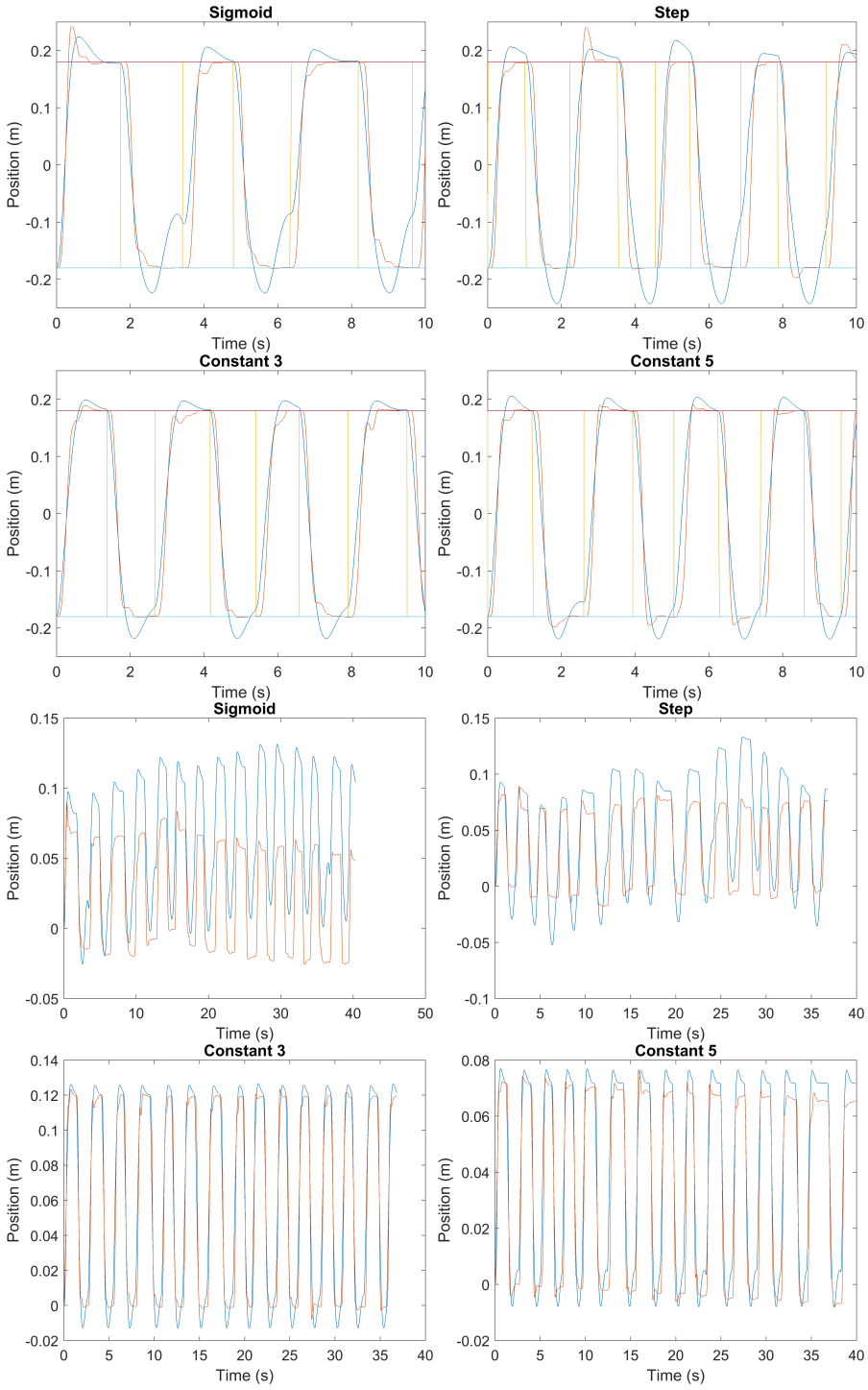

Fig. 9. Top: Time Series of pointer position predicted by the model (blue), as well as the actual pointer position (red) for an average user. The target position is shown for orientation (yellow). Bounds of the targets are shown as horizontal colored lines (ID 8). The model can predict the position over time of the pointer rather well. However, no apparent differences in the dynamic behavior with the four transfer functions are visible in the time series. **Bottom:** Time Series of mouse position predicted by the four models (blue), as well as the actual mouse position (red) for one average user. It can be seen that the mouse of the user (red) is drifting considerably when using a PA function, but not with a constant gain. The reason for this is that users reach different velocities when moving left vs. right. Because the PA function increases the gain as a function of velocity, the mouse is drifting into the direction where higher velocities are reached. This effect is stronger for the sigmoid PA function than for the step PA function.

6.4 System Identification Process

The 2OL model has three free parameters for each direction (mass $1/k_1$, spring k_3, damping k_2). These parameters need to be adjusted to fit the experimental data. This process is called *System Identification*. There exists a comprehensive literature on System Identification with a wealth of methods specialized for identifying specific classes of systems [15].

Because the transfer functions introduce a nonlinearity in the model, the parameters are difficult to identify using classical system identification techniques. Therefore, we resorted to a very general technique based on simulation and optimization. In this technique, an optimizer aims to identify the parameter set that minimizes the error between the model and experimental data. The objective function determines which aspects of user behavior the model is fit to. As objective function we use the sum of sum squared errors (SSE) between actual pointer position, velocity and acceleration, and simulated pointer position, velocity, and acceleration, over time.

As optimizer, we used the Simplex search method as implemented in the Matlab fminsearch function.

We split the experimental data into a training and a test set. For each PA function, we use the first half of trials for training, and the remaining half of trials for evaluation.

7 Results

7.1 Model Accuracy

Figure 9 shows the actual and predicted pointer position over time. It is clear that regardless of the PA function used, our models are able to predict the pointer position over time rather well. However, in order to detect differences in the dynamic behavior between different PA functions, velocity and acceleration of pointer position must be investigated. Figure 8 shows the root mean square error (RMSE) between pointer position, velocity, and acceleration over time as predicted by the model and observed from the participants. The models do not predict acceleration using the Step PA function well. This might be an artifact of our evaluation procedure. For the user, acceleration is calculated from a polynomial approximation of pointer velocity, smoothing out discontinuous acceleration spikes. For the model, pointer acceleration is calculated directly.

7.2 Model Parameters

Table 1 shows the model parameters identified from the pointing data through the optimization process. When interpreting the model as a spring-mass damper system, the parameter k_1 can be interpreted as the inverse of the mass (e.g., of the arm) $1/m$. Similarly, parameter k_2 can be interpreted as the damping, and k_3 as the spring pulling the arm to the target. The damping does not seem to be changed between left and right movements. The lower constant gain 3 is counterbalanced with lower mass ($m = 1/k_1$) compared to gain 5. This can be interpreted as the model adjusting to the PA function and changing its parameters accordingly. Similarly, we can expect users to change their behavior as we change the PA function.

PA Function	Sigmoid	Step	Constant 3	Constant 5
k1l	2.59(0.33,5.93)	2.77(0.33,5.86)	3.26(0.39,9.64)	2.71(0.48,6.12)
k1r	5.78(2.78,14.06)	5.49(3.49,13.17)	5.42(3.37,11.33)	5.14(2.55,8.91)
k2l	0.28(0.11,0.43)	0.26(0.11,0.41)	0.29(0.13,0.44)	0.28(0.13,0.39)
k2r	0.29(0.15,0.47)	0.28(0.17,0.45)	0.36(0.22,0.47)	0.32(0.20,0.43)
k3l	1.90(0.97,4.18)	1.91(0.95,4.30)	1.92(0.89,5.45)	2.20(0.88,5.23)
k3r	1.04(0.76,1.42)	1.10(0.76,1.46)	0.99(0.80,1.61)	0.92(0.77,1.12)

Table 1. Identified Model Parameters. Given are mean(min,max) for each PA Function.

7.3 Mouse Drift

Movements in different directions achieve different velocities. This creates problems when PA functions are used. Movement in the faster direction gets amplified by higher gains, which results in less distance covered by the physical mouse. When the user is moving the pointer back to the original position at lower speed, the mouse needs to cover more space because of the lower gain of the PA function. Over time, the mouse is thus drifting, creating the need to clutch because of the limited workspace. This behavior gets replicated by the model. Figure 9 shows the actual mouse position (red), as integrated from mouse displacements, and the mouse position predicted by the model (blue) over time. It can be seen that with static gain, neither actual nor predicted mouse positions drift. With both PA functions, both actual and predicted mouse position drift considerably. The drift is larger for the Sigmoid function, because differences in gain are integrated over more different velocities, and therefore more time. It should be noted that the model is not trained on the mouse movements, but only on the resulting pointer movements. Thus, mouse drift is an emergent property of the model and does not exactly replicate user mouse drift.

7.4 Phase Space Behavior

Figure 10 shows the phase space plots for the two PA functions and two constant gains. In these plots, pointer velocity is plotted over pointer position. For constant gain, a symmetric phase space plots (e.g., almost half-circle) indicates an open-loop movement, in which acceleration and deceleration are equally steep. During open-loop movements, visual feedback does not influence the motion. Closed-loop movement, in which the user adapts the deceleration phase according to visual feedback, is often characterized by asymmetric phase space plots (e.g., half-egg shaped). The acceleration is steeper, while the deceleration more gradually homes in on the target. The control model presented in this paper represents closed-loop control. In the above plots it becomes clear that PA functions introduce distortions in phase space plots that do not appear in physical (constant gain) behavior. In particular the step PA function contains discontinuities that do not appear otherwise. The center section of the plot is practically “lifted up” where the velocity crosses the threshold. This effect is also replicated by the model.

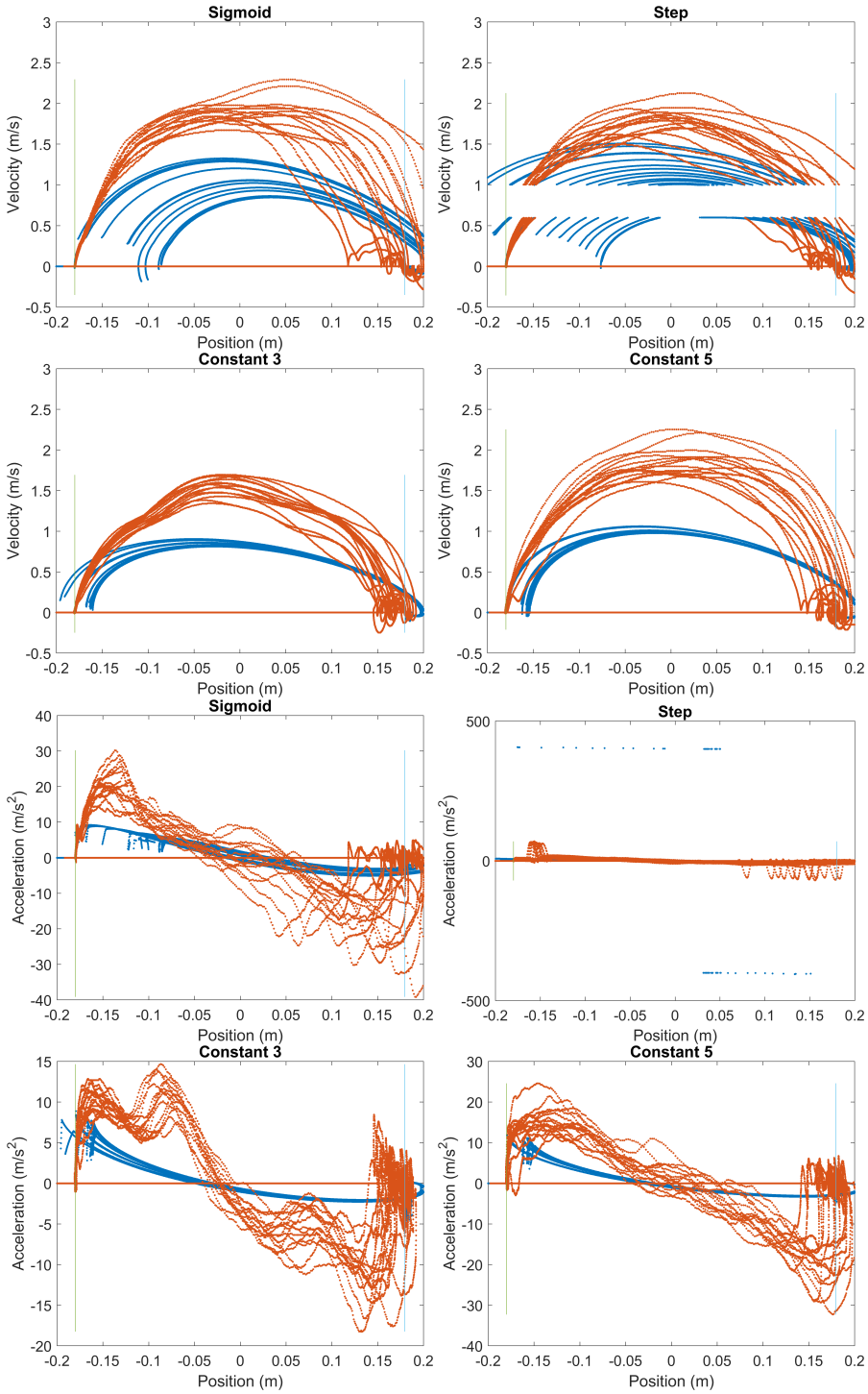

Fig. 10. Top: Phase space plot of pointer velocity predicted by the models (blue), as well as phase space plot of the actual pointer (red) for one average user. For the Step TF, there is a jump in velocity when the velocity crosses the step both during acceleration and deceleration. This phenomenon is replicated by the model (blue). **Bottom:** Hooke plots of pointer acceleration predicted by the model (blue), as well as the actual pointer acceleration (red). The reached acceleration is higher than for constant gain. The Step TF results in steep acceleration and deceleration spikes when the velocity crosses the step. This behavior is replicated by the model. For constant gain 3, this user shows two acceleration spikes per movement. One hypothesis might be that this user starts to accelerate the mouse, but only overcomes the static friction of the arm on the table slightly later, resulting in a second acceleration spike.

7.5 Hooke Plots

Figure 10 shows the Hooke portraits of all transfer functions. In these plots, pointer acceleration is plotted over pointer position. It can be seen that the user Hooke portraits are approximately N shaped, which is indicative of the high ID used in the task [3]. With low ID, one could expect a more backslash (\) shaped Hooke plot, indicating oscillatory behavior. While the user Hooke plot is relatively symmetric N shaped, the model deviates from the user by a more asymmetric shape. The momentary increase in acceleration is indicative of the second order of the model (two chained integrators), where the model can switch acceleration (but not velocity or position) instantaneously, e.g., when the target switches. The acceleration then falls gradually as the pointer approaches the target. The most obvious impact the Step PA function has are the discontinuous acceleration and deceleration spikes when the velocity crosses the step. The model replicates these spikes, however in different positions than the user, because the model underestimates pointer velocity (see Figure 10).

8 Discussion

Our model can predict pointer position over time with some accuracy, while there are systematic deviations from user behavior in velocity and especially acceleration. More importantly, it qualitatively reproduces phenomena that appear in human pointing behavior with pointer acceleration functions. These include the discontinuous phase space plot and acceleration spikes in Hooke plot when using Step PA functions. Also the mouse drift when using PA functions is explained and predicted qualitatively, although not quantitatively (the quantity of mouse drift differs between model and user).

There exist numerous opportunities for improving the model. In particular, the systematic deviations from user behavior regarding velocity and acceleration behavior warrant further investigation. Perhaps most importantly for the case of simulation of PA functions, the model systematically underestimates user velocities. This will result in the application of PA gain at different moments compared to the user. For example, in Figure 10, we see that the Step PA function switches gain at different moments from the actual data. This and other issues should be addressed in future work.

More generally, modeling and simulation of pointing dynamics is important for HCI for a variety of reasons. First, the phenomena of pointing dynamics are very fast (in the order of milliseconds), so that they are less accessible to human intuition than slower phenomena in interaction. This might also explain why pointing dynamics has not received very much attention in HCI to date. Second, many inputs and outputs of the human-computer system are not be easily accessible, or the phenomena of interest are obscured by noise. For example, due to the noisy mouse sensor, mouse and pointer acceleration are difficult to investigate, because phenomena are obscured by noise in mouse motion data. Third, there is considerable variability in human pointing dynamics. In order to detect patterns and phenomena in dynamics, many trials are necessary, increasing the cost of experiments. Simulations of pointing dynamics can be used in an exploratory way to predict phenomena that can then be validated or refuted by experiments.

Computational modeling of PA functions and their simulation might ultimately lead to the ability to computationally optimize PA functions and other transfer functions. This might ultimately lead the field of HCI to conducting more simulations of interaction. However, given the model developed in this paper, we are still quite far from that vision, and it can be seen merely as a first step towards this long-term direction.

At the current step, the model can be seen as more of a tool for thought and inspiration. It shows that we can computationally understand and simulate the effect of PA functions on pointing. In particular, it provides researchers, practitioners and students a concise way of thinking about PA functions. This might help to understand the way PA functions influence the pointing process more deeply. It provides a common vocabulary and conceptual framework for discussing and integrating knowledge about phenomena related to PA functions. It can help in the generation of research questions and hypotheses, and conceptually guide the design of PA functions. We hope that control theoretical modeling of human-computer interaction will become an important research direction for HCI in the future.

9 Conclusion

We have proposed a control theoretical model of pointing dynamics using PA functions. This model allows us to simulate and quantitatively predict pointer position, velocity and acceleration over time and space using PA functions with some accuracy. More importantly, it qualitatively reproduces and provides an explanation for many phenomena that occur in pointing dynamics using PA functions. These include the discontinuous phase space plots and isolated acceleration spikes in Hooke plots when using Step PA functions. Also mouse drift when using PA functions is computationally explained. More importantly, the model provides a tool for thought and a consistent vocabulary for thinking and talking about PA functions.

References

1. Barrett, R.C., Selker, E.J., Rutledge, J.D., Olyha, R.S.: Negative inertia: A dynamic pointing function. In: Conference Companion on Human Factors in Computing Systems. pp. 316–317. CHI '95, ACM, New York, NY, USA (1995), http://doi.acm.org/10.1145/223355.223692
2. Beamish, D., Bhatti, S.A., MacKenzie, I.S., Wu, J.: Fifty years later: a neurodynamic explanation of fitts' law. Journal of The Royal Society Interface 3(10), 649–654 (2006), http://rsif.royalsocietypublishing.org/content/3/10/649
3. Bootsma, R.J., Fernandez, L., Mottet, D.: Behind fitts law: kinematic patterns in goal-directed movements. International Journal of Human-Computer Studies 61(6), 811–821 (2004)
4. Bullock, D., Grossberg, S.: Neural dynamics of planned arm movements: emergent invariants and speed-accuracy properties during trajectory formation. Psychological review 95(1), 49 (1988)
5. Casiez, G., Roussel, N.: No more bricolage!: Methods and tools to characterize, replicate and compare pointing transfer functions. In: Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology. pp. 603–614. UIST '11, ACM, New York, NY, USA (2011), http://doi.acm.org/10.1145/2047196.2047276

6. Casiez, G., Vogel, D., Balakrishnan, R., Cockburn, A.: The impact of control-display gain on user performance in pointing tasks. Human–computer interaction 23(3), 215–250 (2008)
7. Crossman, E.R.F.W., Goodeve, P.J.: Feedback control of hand-movement and fitts' law. The Quarterly Journal of Experimental Psychology 35(2), 251–278 (1983)
8. Fitts, P.M.: The information capacity of the human motor system in controlling the amplitude of movement. Journal of Experimental Psychology 47(6), 381–391 (1954)
9. Fitts, P.M., Peterson, J.R.: Information capacity of discrete motor responses. Journal of experimental psychology 67(2), 103 (1964)
10. Frees, S., Kessler, G.D., Kay, E.: Prism interaction for enhancing control in immersive virtual environments. ACM Transactions on Computer-Human Interaction (TOCHI) 14(1), 2 (2007)
11. Gallo, L., Minutolo, A.: Design and comparative evaluation of smoothed pointing: A velocity-oriented remote pointing enhancement technique. International Journal of Human-Computer Studies 70(4), 287–300 (2012)
12. Jagacinski, R.J., Flach, J.M.: Control Theory for Humans: Quantitative approaches to modeling performance. Lawrence Erlbaum, Mahwah, New Jersey (2003)
13. Jellinek, H.D., Card, S.K.: Powermice and user performance. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 213–220. CHI '90, ACM, New York, NY, USA (1990), http://doi.acm.org/10.1145/97243.97276
14. König, W.A., Gerken, J., Dierdorf, S., Reiterer, H.: Adaptive pointing–design and evaluation of a precision enhancing technique for absolute pointing devices. In: IFIP Conference on Human-Computer Interaction. pp. 658–671. Springer (2009)
15. Ljung, L.: System Identification — Theory for the User. Prentice Hall, Englewood Cliffs, New Jersey, USA (1987)
16. MacKenzie, I.S.: Fitts' law as a research and design tool in human–computer interaction. Human-computer interaction 7(1), 91–139 (1992)
17. Meyer, D.E., Smith, J.E.K., Kornblum, S., Abrams, R.A., Wright, C.E.: Speed-accuracy trade-offs in aimed movements: Toward a theory of rapid voluntary action. M. Jeannerod (Ed.), Attention and Performance XIII pp. 173–226 (1990)
18. Nancel, M., Pietriga, E., Chapuis, O., Beaudouin-Lafon, M.: Mid-air pointing on ultra-walls. ACM Transactions on Computer-Human Interaction (TOCHI) 22(5), 21 (2015)
19. Poupyrev, I., Billinghurst, M., Weghorst, S., Ichikawa, T.: The go-go interaction technique: non-linear mapping for direct manipulation in vr. In: Proceedings of the 9th annual ACM symposium on User interface software and technology. pp. 79–80. ACM (1996)
20. Rutledge, J.D., Selker, E.J.: Force-to-motion functions for pointing. In: Proceedings of the IFIP TC13 Third Interational Conference on Human-Computer Interaction. pp. 701–706. North-Holland Publishing Co. (1990)
21. Schmidt, R.A., Lee, T.: Motor control and learning. Human kinetics (1988)
22. Shannon, C.E., Weaver, W.: The mathematical theory of communication. University of Illinois press (2015)
23. Sheridan, T.B., Ferrell, W.R.: Man-machine Systems: Information, Control, and Decision Models of Human Performance. M.I.T. Press, Cambridge, USA (1974)
24. Varnell, J.P., Zhang, F.: Characteristics of human pointing motions with acceleration. In: Proc. of IEEE 54th Annual Conference on Decision and Control (CDC2015). pp. 5364–5369. Osaka, Japan (2015)