# A mouse adapter for people with hand tremor

J. L. Levine
M. A. Schappert

Hand tremor, which affects millions of individuals worldwide, can make it difficult or impossible to operate computers that rely on a mouse, or similar pointing device, for controlling the user interface. We describe an assistive adapter that, when inserted between the mouse and the computer, provides digital motion-smoothing filtering, rejection of inadvertent mouse button clicks, and enhanced double clicking. Because its behavior closely emulates a standard mouse, this setup is operating-system independent and requires no special software on the computer. Its assistive features are active for any application with a mouse-driven interface. In a preliminary test involving people with essential tremor, most subjects reported improvements ranging from moderate to considerable.

## INTRODUCTION

The project we describe here began soon after we attended Information Technology for Seniors, a workshop organized by the IBM Academy of Technology in 2002. The IBM Academy of Technology is a body of technical experts whose mission is to advise the company executives on technical issues and to facilitate communication among the various technical groups within IBM.

At the workshop we learned that physical problems related to aging frequently include loss of visual acuity, reduced hearing, and hand tremor, all of which can impede computer access. We also learned that, whereas products are commercially available for people with vision and hearing problems, little is available to assist those with hand tremor. We then began a modest program to investigate whether digital filtering of the mouse data stream could provide a solution to this problem. Our intention was not to do research on the symptoms of hand tremor, but rather to empirically develop assistive equipment that enables a substantial fraction of those with hand tremor to use a computer mouse.

Through a joint study agreement, we joined forces with the assistive technology group led by Dr. Cathy Bodine at the University of Colorado Health Science Center.[1] This group, which works routinely with people having a variety of mobility and perceptual problems, helped us conduct a number of preliminary tests. These tests, in which we used special-purpose DOS-based test programs with several types of filters, indicated that filtering could be helpful and

©Copyright 2005 by International Business Machines Corporation. Copying in printed form for private use is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the Journal reference and IBM copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied or distributed royalty free without further permission by computer-based and other information-service systems. Permission to republish any other portion of the paper must be obtained from the Editor. 0018-8670/05/$5.00 © 2005 IBM

encouraged us to find a method that would work for any application with a mouse-driven interface. The tests also uncovered several problems that affect the selection of tremor-smoothing technology and its implementation. The high degree of temporal variability exhibited by many people with tremor and their limited computer literacy were of particular concern. This combination made it difficult to provide an optimal solution for all individuals, and eventually led us to a hardware implementation with very simple controls that was effective for most people with hand tremor.

■ For motion-smoothing filtering we selected a low-pass digital filter primarily because it could be easily adjusted by the user with a single control ■

Many people develop some form of tremor, an involuntary shaking of a body part. When it affects a person's hands, tremor can interfere with many of the activities of daily living that require steady hands, such as the use of a computer mouse or other pointing device. Tremor can be caused by an injury, by an illness such as Parkinson's disease, by hereditary factors, and by aging. When caused by hereditary factors or by aging, it is referred to as essential tremor (ET), and it affects perhaps 15 to 20 percent of people over the age of 65.[2] Frequencies of tremor movements typically range between 4 and 12 Hz (cycles per second).[3] When tremor is associated with an illness or injury, it is often more severe than ET and may be accompanied by a loss of overall control of the hand. In this case, the use of a mouse-driven interface is very problematical. If the level of hand tremor is moderate, which is the case for at least a subset of people with illness-related hand tremor, the problem can be alleviated by applying a suitable digital filter to the data stream from a standard mouse. Such a solution would be effective for the large number of older people who nowadays use computers routinely for activities such as e-mail, word processing, and browsing a library catalog.

Some excellent work has been done in the area of tremor filters, including development of narrow-band rejection filters[4] and adaptive finite impulse response filters.[5] In addition, some very interesting work has been done using haptic (force generating) mice to implement "gravity wells" around selected regions of the screen representing buttons.[6] However, the filters or gravity wells were built into programs designed to optimize or test the filters, or to use them for a specific purpose such as micro-surgery. It is extremely difficult and costly to do this for general-purpose programs, such as Internet browsers, word processors, and image processors. Moreover, modifying these programs to track changes in, say, HTML (Hypertext Markup Language) or word-processor formats, would be a major challenge. However, this can be avoided. For Microsoft Windows**-based systems, mouse input is handled by the operating system, which, via a complex set of device drivers, positions the cursor and passes "mouse events" (changes in location and button state) to the running programs. Therefore, if a filter is applied to the mouse data stream in such a way that the operating system receives only smoothed mouse data, all programs will work without modification.

There are several design alternatives for a filtering mechanism. Filtering can be done in the mouse, in an adapter inserted between the mouse and the computer, or in the computer. The last choice involves attaching a special device driver to the system device driver that handles the raw data from the mouse. As we will show, there are several considerations that favor the use of an adapter. Our approach is to place a digital smoothing filter in the data path between the mouse-motion sensors and the operating-system software component that calculates the cursor coordinates. If this is done properly, all application programs that use mouse input will work without modification.

The rest of the paper is organized as follows. In the next section we describe the considerations that affected our design, including the nature of tremor, customizing the behavior of the adapter, alternatives for the placement of controls, and filtering requirements. In the section that follows, we describe the implementation of the adapter, which involves hardware and firmware components. In the closing section we discuss results and plans for the future.

## DESIGN CONSIDERATIONS

In this section we discuss the major factors that impacted the design of the adapter.

## The nature of tremor

As noted earlier, tremor is characterized by a high degree of variability. Frequently, a person is free of symptoms until a task has to be performed, especially one requiring fine motor control (this type of tremor is sometimes referred to as "intention tremor"). Tremor may also start or stop during a task, sometimes for no apparent reason. This variability explains in part the lack of accurate statistics for the number of people afflicted by it (estimates for the U.S. range from 1 to 20 million).

The variability of tremor also makes it difficult to customize the filtering mechanism as a one-time setup task. An ideal solution for the variability problem would consist of an adaptive filter that is continuously adjusted so as to minimize the discrepancy between the desired and the actual mouse-driven cursor motion. This could be done when the desired cursor path is known, as when a specialized application is run. For example, the user can be asked to trace a curve displayed on the screen or pursue a moving target.[5] However, if the user is doing normal productive work while the customizing program runs in the background, there is no obvious way to calculate the intended mouse path, especially in a multitasking operating system where the mouse events may be passed to different programs in response to cursor position and mouse button or keyboard activity. We note further that there are many activities, such as sketching with a mouse, in which the intended cursor path only exists in the user's mind, and may be indistinguishable from noise. One interesting possibility, which we have not explored, is the fuzzy logic filter used in a joystick controller for piloting wheel chairs.[7]

## Customizing the adapter behavior

We considered the possible use of a program that customizes the filtering mechanism to match the user's needs. We found this solution impractical for several reasons. As noted above, tremor behaves unpredictably and may not be present when the user is trying to run the setup program. Even if the setup program runs successfully, the tremor intensity changes over time, thus requiring frequent readjustment. Many casual computer users lack the necessary computer skills and the self confidence necessary to run a setup program by themselves, especially if there are options to select or parameters to adjust. Although many casual computer users are able to use specific applications, such as e-mail, with great competence, they are unaware of the many Windows features, such as the control panel, that can be used to change mouse and keyboard properties. The user may find it difficult to interact with such a program if tremor is present. Finally, use of such a program would be impractical for users of public terminals.


Figure 1
A mouse adapter for people with hand tremor

## Unintended button clicks

We found during our initial trials that tremor sometimes caused unintended mouse button clicks, and made double clicking extremely difficult. Most mice are designed so that the user's fingers rest on the mouse buttons. A button can be clicked accidentally by a slight finger tremor. These false clicks are suppressed by applying a simple timing algorithm, in which the change to button state is only passed through to the computer if it persists for a preset time interval. The algorithm is described in Reference 8. *Figure 1* shows the controlling 3-position toggle switch (labeled Button Delay) that selects between "Off" and two hold-time values, 100 milliseconds (low) and 125 milliseconds (high), respectively.

## Double click support

The interpretation of a double click, pressing the mouse button twice in quick succession, is typically different from that of a single click. Whereas a single click might, for example, select a file, a double click would open that file. A valid double click consists of two separate button-down events, with a maximum allowed time and mouse motion between the two events. These are set by two operating-system parameters. Depending on the parameter values, activating a double click may be difficult for some

people, even for those without hand tremor. It is possible to relax both conditions using the operating-system interface, but few people are aware of this. Also, any such adjustment may apply to all people who use the computer. One can configure Windows so that double clicking is not required, but the other users of the computer may find this annoying. Therefore, an algorithm was devised that recognizes a double click under "relaxed" conditions, that is, for suitably long intervals between the pair of clicks. It then moves the cursor back to its position at the first button-down event, and finally sends a perfect double click signal to the computer. The algorithm is described in Reference 8. Figure 1 shows the controlling 3-position toggle switch (labeled Double Click) that selects between "Off" and two relaxed choices of the allowed motion and time between clicks.

## Filter design

We make use of a filter with a single, easily adjusted parameter. After some experimentation, we settled on a simple low pass filter whose amplitude versus frequency response $A(f)$ is given in equation 1:

$$A(f) = 1/\sqrt{1 + (f/f_c)^2} \tag{1}$$

Here, the adjustable parameter $f_c$ is the frequency at which the filter response has fallen to 0.707 of the zero-frequency response. Such a filter passes slow intentional motions, while progressively attenuating the high frequency tremor components above $f_c$. The parameter $f_c$ can be adjusted with a simple control, as desired. The user does not require any knowledge as to what is actually being adjusted. As shown in Figure 1, we use a physical knob labeled "Filter Setting."

An important requirement is a means to easily disable the filter when it is not needed, because filters introduce time delays which may be annoying and may affect eye-hand feedback. Providing a simple and clearly marked on/off switch is particularly important if the computer is shared with other people, and essential if the computer is located in a public facility. In Figure 1, the switch is labeled "Assistance."

As noted above, a simple, low-pass digital filter was selected primarily because it could be quickly adjusted by the user with a single control. Such a filter would normally accept as input integer values representing incremental mouse motion, and provide as output a sequence of exponentially decreasing real values, not necessarily integers. Because the mouse protocol requires that integers be sent to the computer, a special algorithm was developed to deal with this problem by keeping careful track of the fractional remainders. The algorithm is described in Reference 8.

A PS/2* mouse interrupts the computer's CPU each time it transmits a byte. In order not to generate unnecessary interrupts, it only transmits when the mouse moves or a button changes state. As a result, calls to the filter subroutine cease abruptly when mouse motion ends. This has two undesirable consequences. First, because the normal response of this type of filter after mouse motion ceases is an exponentially decreasing cursor motion, the abrupt termination of the motion is annoying to the user. Second, when mouse motion resumes, the exponential decay from the previous cycle is now added to the new motion, a behavior that is distracting to the user, especially if the new motion is in a different direction from the previous one. The solution involves a hardware timer that is part of the adapter's P2 microprocessor (***Figure 2***). The timer is used to generate a predefined number of pseudo mouse events at the correct transmission rate. The number is calculated to be sufficient to produce the normal exponential decay.

Another complication is caused by the nonlinear transformation applied to the motion increments by the operating system's internal software, which emphasizes large increments over small ones. This is done to allow the cursor to be positioned to single-pixel precision, while also allowing full-screen motion with moderate mouse motion. However, when a large motion increment occurs, the filter reduces it to a series of smaller, exponentially decreasing increments. The nonlinear transformation applied to these small increments has the effect of reducing the gain of the filter by an amount that depends on the degree of filtering. This is corrected by including a multiplication factor that depends on the degree of filtering.

Finally, the slow exponential decay of cursor motion after the mouse stops can be annoying when filter cutoff frequency is very low. Therefore, the filter algorithm was modified to reset the filter a predetermined time after mouse motion ceases.

### Placement of controls

The filter controls that can be provided depend on where the filter is located. If the filtering takes place in the internal microprocessor of the mouse, the required switches and knobs can be placed in the mouse housing, perhaps behind a flip-up door. Unfortunately, the small physical size of a standard mouse would make these controls small and hard to use, especially for people with tremor. Further, it would be uneconomical to rely on specialized mouse devices of different shapes, sizes, and features (e.g., scroll wheels, scroll sticks, extra buttons) to suit different preferences. For these reasons, the idea of a specialized mouse was dropped from consideration.

The filtering mechanism could be located in the mouse device driver, in which case an on-screen interface would be used. Many people find such an interface non-intuitive, even intimidating. Furthermore, people with tremor may find controlling an on-screen interface challenging. Although adjustments could be made by using the keyboard, this is for many an even less intuitive interface. We experimented with such a device driver-based filter (such a filtering mechanism could be packaged and distributed as a free or low-cost utility) and found that of the people who used both the device-driver solution and the separate adapter, most preferred the separate adapter. In addition, device drivers require long-term software support because they are sensitive to changes in the operating system and the hardware. Some mouse device drivers, installed when a new mouse device is acquired, may disable the filter. The current version of the device driver works well for many standard mice, provided the device drivers are the ones supplied with the operating system. Although an installation and removal utility has been developed, it does not yet handle certain hardware configurations, for example keyboards that include built-in pointing devices. Some problems may occur if the driver is installed and removed multiple times. We have no current plans to develop this into a software product, but a pre-beta-level version for Windows 2000 and Windows XP** is available for an at-risk trial.[9]

### Final configuration

Our solution involves an external adapter to be inserted between the mouse and the computer. The adapter interface to the computer is the standard mouse interface. Adjustments are enabled through familiar switches and knobs located on the adapter box. These can be large and conveniently spaced, at least for desktop use. Further, one can easily adjust the filter with one hand while moving the mouse with the other, with instant visual feedback.

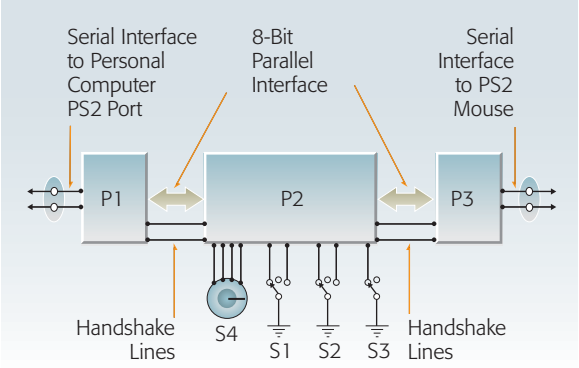


**Figure 2**
Block diagram of the adapter circuit

## IMPLEMENTATION

A prototype of our microprocessor-based adapter is shown in Figure 1. The adapter, which is connected between a PS/2 mouse and a PS/2 computer port, closely emulates a PS/2 mouse (but see the comments on laptop computers in the section "Results and conclusions"). The latest version of this adapter has a knob to adjust the degree of filtering, a toggle switch to suppress unintended button clicks, another toggle switch to control the enhanced double-clicking feature, and a third toggle switch that disables all the accessibility features when they are not needed.

### Adapter hardware

The hardware configuration shown in Figure 2 consists of three low-cost RISC (reduced instruction set computer) processors, labeled P1, P2, and P3, and four switches, S1 through S4. Power is provided through the computer PS/2 port.

Alternative configurations, firmware flowcharts, and details of the algorithms can be found in a patent held by the authors; however, the hardware and the key algorithms are essentially the same as described in the patent.[8]

An industry-standard PS/2 mouse communicates with a PC using a slow, 10 KHz two-wire serial

command-and-response protocol[10] developed by IBM in the late 1980s. Referring to Figure 2, processor P1 mediates data transfers in either

■ The filtering mechanism and its controls are located in a separate adapter that is inserted between the mouse and the computer ■

direction between the PC and processor P2. It uses the two-wire serial protocol to transfer data to and from the PC, and a fast, interrupt-driven-transfer 8-bit parallel protocol with processor P2. Similarly, processor P3 mediates data transfers between processor P2 and the mouse. Processor P2 implements the assistive features. Considerable effort was expended to make the interaction with the computer follow the IBM protocol, so that the adapter plus mouse would appear to the operating system as a standard mouse.

P2 is more powerful than P1 and P3, having an instruction set suitable for implementing a digital filter and running the algorithms needed to provide the button-assist features mentioned earlier. Control of the button features is provided by three-position toggle switches S1 and S2. Two-position toggle switch S3 is used to disable all the features when not needed. The degree of filtering is controlled by the 16-position binary-encoded rotary switch S4. The adapter firmware converts the rotary-switch setting to a filter-time constant in the range of 50 to 950 milliseconds, using a nonlinear lookup table. This corresponds to cutoff frequencies $f_c$ in the range 0.17 Hz to 3 Hz. The range was chosen to preferentially remove tremor frequencies which, as noted earlier, are typically in the range 4–12 Hz.[3] The components, including the switches, are soldered to a small printed circuit board that is housed in a plastic box of dimensions $3 \times 4 \times 1$ inches. The dimensions were chosen to allow switches of reasonable size and spacing. Cables connect the adapter to the computer and to the mouse.

### Adapter firmware

The original firmware supported the basic, two-button mice. In the latest version of the adapter, the firmware was extended to accommodate mice with scrolling mechanisms and up to five buttons. The firmware for the P2 processor, which implements the assistive features, turned out to be surprisingly complex, in part because of the ad hoc way in which the PS/2 protocol was used to handle scroll wheels and other mouse features that were not available when the protocol was defined. We provide here only a brief description.

When a basic two- or three-button mouse is moved, it transmits 3-byte data packets containing X and Y increments and button states. Packets are transmitted at a given rate (the rate is set by the computer through commands sent to the mouse), a typical rate being 100 per second. Mice with scrolling mechanisms or five buttons encode the additional information in a fourth byte. The computer and mouse establish the packet size and the presence of special features in a special power-up initialization procedure that is not part of the original PS/2 architecture. In this procedure, the computer sends special sequences of standard commands that are recognized by mice equipped with a scroll wheel or extra buttons. The mouse then responds to a "Read Device Type" command with a unique byte, in place of the usual zero.[11] Processor P2 monitors this exchange to determine the packet size. Thereafter, P2 collects the entire 3-byte or 4-byte packets, extracts the X and Y increments, and applies a filter to each increment. The button algorithms are also applied if enabled by the switches. New data packets are then constructed from the modified motion values and button states and sent to the computer in place of the original packets via P1.

## RESULTS AND CONCLUSIONS

A usability test for the adapter was performed with a group of 10 people selected from a support group for people with essential tremor. The group ranged in age from 36 to 79 and included subjects of both sexes. The test included video and audio recordings of the session, a movie-like capture of the screen in which the cursor motion was recorded, and a post-test participant evaluation form. The participants reported moderate to considerable improvement, and all expressed an interest in acquiring the adapter.

The adapter was tested with a variety of mice sold by eight manufacturers. These include native PS/2 mice, as well as Universal Serial Bus (USB) mice equipped with a manufacturer-supplied USB-to-PS/2

adapter. Some were equipped with a scrolling mechanism and had either three or five buttons; the others had either two or three buttons. For these tests we used desktop computers made by three different manufacturers under four versions of Windows (98, ME, 2000, and XP). The adapter performed satisfactorily in all configurations.

It is likely that the PS/2-type ports will be omitted from some or all future desktop models in favor of USB ports. This is already the case with some laptop models. We plan to develop a version of the adapter in which processors P1 and P3 will be replaced by USB-compatible chips.

The built-in pointing devices on laptop computers are not always equipped for connecting with our adapter. The TrackPoint* device used in many IBM ThinkPad* notebook computers is supported by the same microprocessor that handles the PS/2 mouse. In order to provide plug-and-play capability, the communication protocol between the processor and the mouse involves periodic commands from the processor and corresponding responses from the mouse. Unfortunately, these commands interfere with the complex and time-critical data flow through the adapter, leaving it in a nonfunctioning state. This problem can be sidestepped by configuring the TrackPoint to run in a mode in which it is itself disabled when a mouse is connected. Other laptops have not been tested and may not support a similar solution.

About two dozen adapters were built using the original version of the firmware, which supported only two-button mice. The adapters have since been installed on 12 computers located at the State of Colorado Workforce Centers.[12] We recently negotiated a licensing agreement with a small electronics company for manufacturing and marketing the adapters.

## ACKNOWLEDGMENTS

We thank Cathy Bodine and Jim Sandstrum of the University of Colorado for helping test the adapters, Catherine Rice of the International Essential Tremor Foundation for helping identify a manufacturer for the adapters, and Steve Mastrianni for helping implement the device driver-based version of the adapter. Their contributions have been invaluable to the success of the project.

*Trademark, service mark, or registered trademark of International Business Machines Corporation.

**Trademark, service mark, or registered trademark of Microsoft Corporation.

## CITED REFERENCES AND NOTES

1. Assistive Technology Partners, University of Colorado Health Science Center, http://www.uchsc.edu/atp/aboutatp.htm.
2. A useful Web site is http://www.essentialtremor.org.
3. R. J. Elble, "Physiologic and Essential Tremor," *Neurology* **36**, No. 2, 225–231 (1986).
4. C. N. Riviere, R. S. Rader, and N. V. Thakor, "Adaptive Canceling of Physiological Tremor for Improved Precision in Microsurgery," *IEEE Transactions on Biomedical Engineering* **45**, No. 7, 839–846 (1998).
5. J. G. Gonzalez, et al., "A Customized Optimal Filter for Eliminating Operator's Tremor," *Proceedings of SPIE Volume 2590: Telemanipulator and Telepresence Technologies II*, M. Salganicoff, Editor, International Society for Optical Engineering (December 1995), pp. 131–142.
6. S. Keates, P. Langdon, J. Clarkson, and P. Robinson, "Investigating the Use of Force Feedback for Motion Impaired Users," *Proceedings of the 6th ERCIM Workshop, European Research Consortium for Informatics and Mathematics*, Sophia Antipolis, France (2000), pp. 207–212.
7. B. van der Zwaag, D. Corbett, and L. C. Jain, "Minimizing Tremor in a Joystick Controller Using Fuzzy Logic," *Proceedings of the Third International Conference on Knowledge-Based Intelligent Information Engineering Systems*, 31 Aug–1 Sept 1999, Adelaide, Australia, IEEE, New York (1999), pp. 5–8.
8. J. L. Levine and M. A Schappert, *Method and Adapter for Performing Assistive Motion Data Processing and/or Button Data Processing External to a Computer*, U.S. Patent 6,650,313 (11/18/2003).
9. Mouse Smoothing Software, alphaWorks, IBM Corporation, http://www.alphaworks.ibm.com/tech/mousesmoothing.
10. *PS/2 Mouse Technical Reference*, Publication S68X-2229-00, Second Edition, IBM Corporation (June, 1989).
11. *Windows and the 5-Button Wheel Mouse*, Microsoft Corporation (December 4, 2001), http://www.microsoft.com/whdc/device/input/5b_wheel.mspx.
12. Colorado Workforce Center, State of Colorado, http://www.yourworkforcecenter.com/.

*Accepted for publication January 7, 2005.*
*Published online July 22, 2005.*

*James L. Levine*
*IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598 (levine2@us.ibm.com)*. Dr. Levine is a research staff member in the Physical Sciences Department at the Thomas J. Watson Research Center. He received a B.S. degree in physics at the Massachusetts Institute of Technology in 1958 and M.S. and Ph.D. degrees in physics at the University of Minnesota in 1960 and 1962, respectively. He joined IBM in 1962 at the

Watson Laboratory in New York City. Dr. Levine is the author of 19 papers on low-temperature and ultra-low-temperature physics, superconductivity, gravity wave detection, and computer technology. He holds 31 U.S. patents on touch screens, eye-controlled computers, presence sensors, and other aspects of interactive computer technology. He is currently working on ways to improve computer accessibility for people with hand tremor.

*Michael A. Schappert*
*IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (schap1@us.ibm.com)*. Mr. Schappert received his B.S. degree in computer science from Union College in 1987 and his M.S. degree in computer engineering from Syracuse University in 2001. He started his career at IBM in 1978, when he joined the Research Division to work on an eye-tracker system. At IBM, Mr. Schappert worked on various projects related to human-computer interfaces, such as eye tracking, touch screens, infrared presence sensors, remote infrared pointing devices, and mouse filters. He holds 11 U.S. patents, with several additional patents pending. Mr. Schappert is a Senior Member of the Institute of Electrical and Electronic Engineers and a member of the Association for Computing Machinery. ■