Behavior Research Methods, Instruments, & Computers
1998, 30 (3), 454-456

A simple graphical technique for assessing timer accuracy of computer systems

BRETT MYORS
Macquarie University, Sydney, New South Wales, Australia

Timing accuracy is of perennial concern to researchers conducting reaction time studies, especially for those using computers that were not designed with millisecond accuracy in mind. Inaccuracies can manifest themselves in the form of quantization of the continuous time data stream. The method proposed here is simply to inspect the graph of the order statistics because quantization is readily apparent in such graphs. The method is illustrated with timing measurements of keypresses on an IBM PC.

The accuracy of timing routines will always be an important practical concern for researchers collecting reaction time (RT) data on computers (Beringer, 1992; Bovens & Brysbaert, 1990; Brysbaert, Bovens, d'Ydewalle, & Van Calster, 1989; Bührer, Sparrer, & Weitkunat, 1987; Creeger, Miller, & Paredes, 1990; Crosbie, 1989; Dlhopolsky, 1988; Emerson, 1988; Granaas, 1989; Graves & Bradley, 1987, 1988, 1991; Heathcote, 1988; Segalowitz, 1987, 1988; Segalowitz & Graves, 1990). Inaccuracies and poor resolution in either the timer or the response device manifest themselves as quantization of the continuous time data stream. For example, it is well known that, on an IBM PC, the standard system time-of-day clock runs at 18.207 Hz (Sargent & Shoemaker, 1995), giving a timing resolution of only 54.925 msec. As such, any time data measured by this clock will be quantized into units that are too large for most RT studies. To overcome this difficulty, various techniques have been developed for achieving millisecond timing accuracy on the PC.

Another source of quantization consists of delays built into the PC keyboard. The keyboard contains an 8048 chip that poles the key lines for presses (Sargent & Shoemaker, 1995). When a keypress is detected, the 8048 "waits a few milliseconds to let the key stop bouncing" (Sargent & Shoemaker, 1995, p. 471). This is the main source of the delay. A further delay stems from the serial transmission of the scan code to the CPU, which takes roughly an additional millisecond. Segalowitz and Graves (1990) reported that the keyboard exhibited a delay of about 15 msec on an IBM PC and 10 msec on an AT. They noted that this value varied with configuration but suggested that the keyboard should be considered to have an effective resolution of only about 15 msec.

As a result of these considerations, many authors advocate careful calibration of PC timing routines against an accurate external clock prior to embarking on an RT study. However, while this is recommended, it can involve attaching the clock to one of the output ports or wiring it in line with the keyboard or PC speaker, and many users may not have access to this level of expertise. Therefore, the purpose of this article is to advocate a simple graphical technique for studying timing data which reveals the quantization, and hence inaccuracy, inherent therein. Researchers can then decide whether the times are accurate enough for their purposes.

The method is simply to inspect the graph of the order statistics; that is, sort the time data, graph them, and look for a smooth continuous function. If the curve is smoothly increasing, this indicates relatively high accuracy; if it looks like a step function, this indicates poor resolution. Often such graphs can strikingly reveal surprising degrees of quantization. The size of the quanta can be readily determined from the step sizes. It may also be useful to trim the top and bottom 5% or 10%, say, of data points before graphing, in order to remove outliers.

Please note that the technique described here is mainly suited for detecting systematic poor resolution in timing routines and response devices. It does not indicate whether all times are displaced by a constant factor, nor will it detect random errors introduced in the time data.[1] The first problem is unlikely to be of concern, unless absolute times are required. For example, if all one wants to do is assess the effect of some treatment, it is irrelevant that the timing data are out a constant factor as long as the two conditions differ significantly. This simply means that the clock is poorly calibrated, not that it is inherently inaccurate. The second source of error will be relegated to the error term if sufficient trials are conducted.

Another warning concerns the use of multitasking environments. It is generally advisable not to use multitasking environments such as Windows or Windows '95 when time-critical data are being collected.

To illustrate the technique, keypress times were measured for reading an alphanumeric keypress and a shift keypress on an IBM-PC–compatible microcomputer.

METHOD

A C program was written to time the getch()[2] function and shift key status on a standard 133-MHz Pentium PC.

Correspondence concerning this article should be addressed to B. Myors, School of Behavioural Sciences, Macquarie University, Sydney, NSW 2109, Australia (e-mail: bmyors@bunyip.bhs.mq.edu.au).

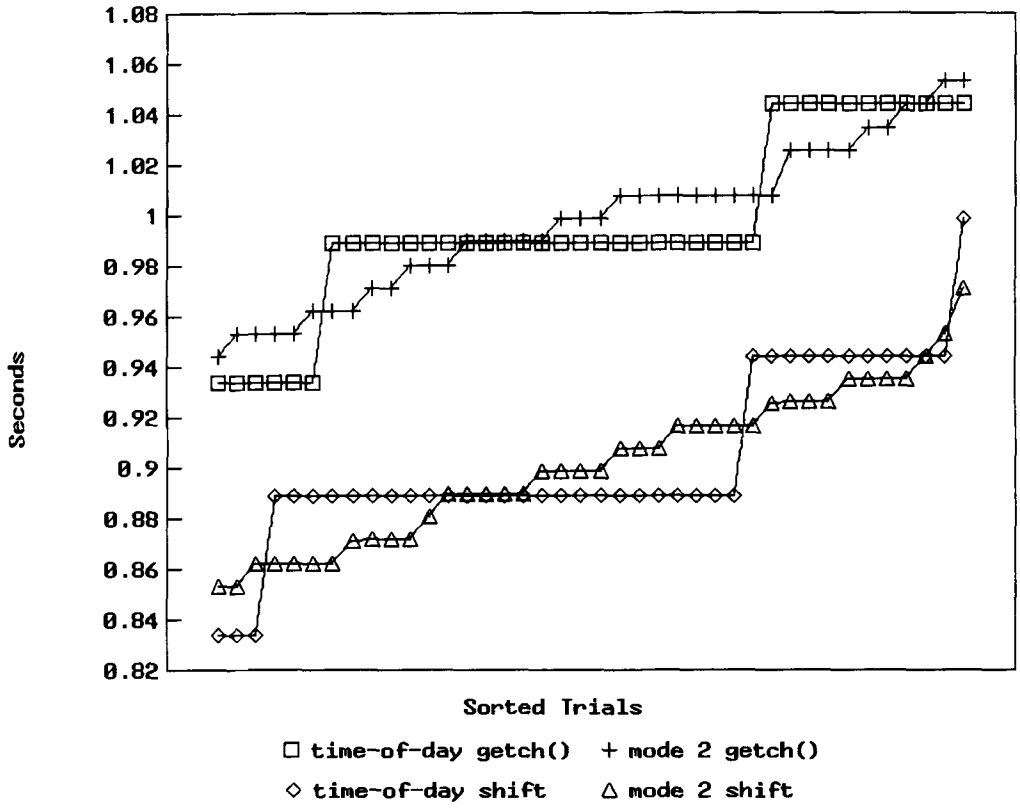

Figure 1. Sorted times of 40 presses of the space bar and the shift key. Trials were conducted with the subject keeping pace with the second hand on a watch. Ten milliseconds have been subtracted from the shift times to separate the curves. In each case, the five shortest and longest times have been trimmed.

Although the BIOS provides a standardized means of reading the shift status, shift status was assessed by reading address 417H of the BIOS data segment directly in order to improve efficiency (see Sargent & Shoemaker, 1995, for details). Use of BIOS calls can introduce further delays. Readers interested in the method can obtain a copy of the program from the author.

Timing consisted of 50 presses of the space bar followed by 50 presses of the shift key. Fifty data points seem to be about the right number for this technique. In each case, the subject was trying to press the keys at a constant rate of 1 press per second, in line with the second hand on a watch.

Two timing methods were investigated. The first involved the low-resolution time-of-day clock already mentioned, and the second used the high-resolution "mode 2 timer" that has a period of 838.1 nsec. The mode 2 timer has been advocated by several authors as the best method of achieving high-resolution timing on the PC (Abrash, 1990; Graves & Bradley, 1991; Sargent & Shoemaker, 1995). The two methods can be used in tandem, since the mode 2 timer involves nothing more than adding the sub 54.925-msec count latched from counter zero of the on-board 8253 programmable interval timer set to mode 2 to the value of the time-of-day clock multiplied by 65,536. Detailed accounts of the mode 2 timer, together with various implementations, can be found in Abrash (1990), Graves and Bradley (1991), and Sargent and Shoemaker (1995). Abrash has also found that the counters on fully compatible 8253 chips are literally stopped during a mode set, thus completely eliminating the rare occurrence of desynchronization between the counter and the time-of-day clock discussed by Graves and Bradley (1991). This method was not used, however, because it relies on an undocumented feature of the 8253 and may not be portable.

RESULTS

Figure 1 shows a graph of the keypress times measured by the two timing methods and illustrates the method. The graph was produced with the Lotus 1-2-3 spreadsheet. Any spreadsheet or statistical package could be used to produce similar graphs. As can be seen, all times are clearly quantized with the times measured by the slow time-of-day clock jumping in larger steps than the fast mode 2 timer. In close agreement with the specified rate of the time-of-day clock, the large steps in the time-of-day

curves are equal to 54.945 msec. In close agreement with Segalowitz and Graves (1990), the steps in the mode 2 curves are in units of about 9.1 msec.

CONCLUSION

Researchers conducting RT studies need to check the accuracy of their measurements very carefully. It is suggested that a graph of the order statistics provides a quick and easy supplement to external calibration. Although the method has been illustrated here with the IBM PC, because the relevant timing information has been well documented, the method is quite general and can be applied to any continuous time data of any duration collected on any computer. It is suggested that researchers who collect continuous time data inspect the graph of the order statistics as an adjunct to reporting the accuracy of their timers.

Availability

Software described in this article is available from the author upon request: bmyors@bunyip.bhs.mq.edu.au.

REFERENCES

ABRASH, M. (1990). *The zen of assembly language*. Glenview, IL: Scott, Foresman.

BERINGER, J. (1992). Timing accuracy of mouse response registration on the IBM microcomputer family. *Behavior Research Methods, Instruments, & Computers*, **24**, 486-490.

BOVENS, N., & BRYSBAERT, M. (1990). IBM PC/XT/AT and PS/2 Turbo Pascal timing with extended resolution. *Behavior Research Methods, Instruments, & Computers*, **22**, 332-334.

BRYSBAERT, M., BOVENS, N., D'YDEWALLE, G., & VAN CALSTER, J. (1989). Turbo Pascal timing routines for the IBM microcomputer family. *Behavior Research Methods, Instruments, & Computers*, **21**, 73-83.

BÜHRER, M., SPARRER, B., & WEITKUNAT, R. (1987). Interval timing routines for the IBM PC/XT/AT microcomputer family. *Behavior Research Methods, Instruments, & Computers*, **19**, 327-334.

CREEGER, C. P., MILLER, K. F., & PAREDES, D. R. (1990). Micromanaging time: Measuring and controlling timing errors in computer-controlled experiments. *Behavior Research Methods, Instruments, & Computers*, **22**, 34-79.

CROSBIE, J. (1989). A simple Turbo Pascal 4.0 program for millisecond timing on the IBM PC/XT/AT. *Behavior Research Methods, Instruments, & Computers*, **21**, 408-413.

DLHOPOLSKY, J. G. (1988). C language functions for millisecond timing on the IBM PC. *Behavior Research Methods, Instruments, & Computers*, **20**, 560-565.

EMERSON, P. L. (1988). TIMEX: A simple IBM AT C language timer with extended resolution. *Behavior Research Methods, Instruments, & Computers*, **20**, 566-572.

GRANAAS, M. M. (1989). TIMEX2: A modified C-language timer for PC AT class machines. *Behavior Research Methods, Instruments, & Computers*, **21**, 619-622.

GRAVES, R. [E.], & BRADLEY, R. (1987). Millisecond interval timer and auditory reaction time programs for the IBM PC. *Behavior Research Methods, Instruments, & Computers*, **19**, 30-35.

GRAVES, R. [E.], & BRADLEY R. (1988). More on millisecond timing and tachistoscope applications for the IBM PC. *Behavior Research Methods, Instruments, & Computers*, **20**, 408-412.

GRAVES, R. E., & BRADLEY R. (1991). Millisecond timing on the IBM PC/XT/AT and PS/2: A review of the options and corrections for the Graves and Bradley algorithm. *Behavior Research Methods, Instruments, & Computers*, **23**, 377-379.

HEATHCOTE, A. (1988). Screen control and timing routines for the IBM microcomputer family using a high-level language. *Behavior Research Methods, Instruments, & Computers*, **20**, 289-297.

SARGENT, M., III, & SHOEMAKER, R. L. (1995). *The personal computer from the inside out* (3rd ed.). New York, Addison-Wesley.

SEGALOWITZ, S. J. (1987). IBM PC tachistoscope: Text stimuli. *Behavior Research Methods, Instruments, & Computers*, **19**, 383-388.

SEGALOWITZ, S. J. (1988). IBM PC tachistoscope: II. Assembly language subroutines. *Behavior Research Methods, Instruments, & Computers*, **20**, 432.

SEGALOWITZ, S. J., & GRAVES, R. E. (1990). Suitability of the IBM XT, AT, and PS/2 keyboard, mouse, and game port as response devices in reaction time paradigms. *Behavior Research Methods, Instruments, & Computers*, **22**, 283-289.

NOTES

1. The author is grateful to an anonymous reviewer for pointing this out.
2. getch() is a standard C function for reading a character from the console.

(Manuscript received December 13, 1996;
revision accepted for publication April 8, 1997.)