Substantiating Programmer Variability

BILL CURTIS

Abstract—**Dickey's critique of the Sackman *et al.* data is well taken. An alternate data set is presented which substantiates the enormous variability in programmer performance. The opportunity for productivity gains and improved experimental methods in research through reducing the range of this variability remains fertile.**

I agree with most of Dickey's [2] comments on the Sackman *et al.* [4] paper. Although the Sackman *et al.* data are not definitive in arguing 20+:1 differences in performance among programmers, other sets of data exist which display such ranges. Below are data my colleagues at GE and I collected two years ago on 54 professional programmers from both military and civilian environments. These data are from the pretest conducted with Experiment 3 (Debugging) reported by Sheppard *et al.* [5]. On this pretest, the programmers were given a modular-sized Fortran program with a simple bug embedded in it. We measured the amount of time they required to find and correct the bug. The first 27 programmers were given a program which was found to be too difficult for the purpose of the pretest, so a different program was given to the second 27. The 27 programmers attempting each program were confronted with an identical task, and performance differences could be attributed directly to differences among programmers in talent, experience, etc. Table I presents distributions of debugging times for each of these two programs.

TABLE I

	Frequency	
Minutes	Program 1	Program 2
1-5		5
6-10	5	10
11-15	7	4
16-20	4	3
21-25	1	1
26-30	1	
31-35	3	
36-40	3	3
41-45	2	
46-50	1	
51-55		
56-60		
61-65		
66-70		1
Range	6-47	3-67

Although the range of performance scores for Program 1 is only about 8:1, the range for Program 2 is about 22:1. This range is not entirely accurate since the unfortunate programmer who spent 67 minutes on this task quit in frustration without discovering the bug. This programmer was not incompetent, however, since he was able to debug the 3 programs involved in the experimental manipulations. Data for 6 other professional programmers involved in this experiment were deleted, since they were unable to debug either the pretest or the experimental programs. In this case exact ratios are not meaningful, but a statement such as "order of magnitude differences in the performance of individual programmers" seems justified. For instance, eliminating the datapoint for 67 minutes from the distribution of times for Program 2 reduced the range ratio to a mere 13:1 (2 programmers required 39 minutes to find the bug).

Sackman's [3] message that substantial performance differences do exist among programmers remains valid. Detecting a 20+:1 range ratio depends upon having one brilliant and one horrid performance in a sample. However, the range ratio is not a particularly stable measure of performance variability among programmers. The dispersions of such data as appear in Table I are better represented by such measures as the standard deviation or semiinterquartile range.

Manuscript received December 8, 1980.

The author is with the Programming Technology Center, International Telephone and Telegraph, Stratford, CT.

Substantial variations in programmer performance can be attributed to individual differences in experience, motivation, intelligence, etc. Thus, important productivity gains could be realized through improved programmer selection, development, and training techniques. These gains would be achieved through eliminating the skewed tails often observed in distributions of programmer performance data. For example, with continued experience on the task the programmer who spent 67 minutes on our pretest improved his performance substantially during later experimental trials.

The gist of my citation [1] of the Sackman *et al.* paper was that differences among programmers are often of sufficient magnitude to disguise performance effects due to software characteristics or practices. I continue to wrestle with this problem in experimental research.

References

[1] B. Curtis, "Measurement and experimentation in software engineering," *Proc. IEEE.*, vol. 68, pp. 1144-1157, Sept. 1980.
[2] T. E. Dickey, "Programmer variability," this issue, pp. 844-846.
[3] H. Sackman, *Man-Computer Problem Solving.* New York: Auerbach, 1970.
[4] H. Sackman, W. J. Erickson, and E. E. Grant, "Exploratory experimental studies comparing online and offline programming performance," *Commun. Assoc. Comput. Mach.*, vol. 11, pp. 3-11, 1968.
[5] S. B. Sheppard, B. Curtis, P. Milliman, and T. Love, "Modern coding practices and programmer performance," *Comput.*, vol. 12, no. 12, pp. 41-49, 1979.

Cascade Configurations for Recursive-Like Adaptive Noise Cancellation

W. A. GARDNER

Abstract—**Two novel recursive-like two-stage adaptive noise cancellers that circumvent the requirement, in prior art one-stage recursive-like cancellers, of incorporating a constraint on filter coefficients, are presented. These novel two-stage cancellers are based on a cascade configuration, rather than the prior art parallel configuration. For applications in which even a small amount of signal distortion is intolerable, a third novel two-stage noise canceller that guarantees distortion-free performance is presented. Finally, a multistage cascade configuration that has the potential for distortion-free, high-performance noise cancellation is presented.**

As explained in [1], constraints imposed on filter coefficients in the recursive-like noise-canceller shown in Fig. 1 can prevent convergence to the best attainable noise canceller. These constraints, which are imposed in order to prevent convergence to the trivial solution $\Phi_p = \Phi_r = 0$, can be eliminated if the configuration shown in Fig. 1 is modified so that Φ_p and Φ_r are not connected in parallel.

One such modification is shown in Fig. 2. In principle, complete noise cancellation would occur if Φ_r were equal to the denominator of the transfer function H/G, and Φ_p were equal to the numerator. However, in practice, when Φ_p and Φ_r are adaptively adjusted (with the LMS algorithm, for example), Φ_r will attempt to approximate the entire rational function H/G, rather than only its denominator (and similarly for Φ_p). Nevertheless this configuration has the potential for outperforming the conventional single-stage canceller, which consists of only the first stage in Fig. 2.

As an alternative the order of the two stages can be reversed as shown in Fig. 3.

Manuscript received January 26, 1981. This work was supported by the Air Force Office of Scientific Research under Grant AFOSR80-0189.

The author is with the Signal and Image Processing Laboratory, Department of Electrical and Computer Engineering, University of California, Davis, CA 95616.