

A NEW ADAPTIVE FILTER SIMULATOR

-by Bernie Hutchins

INTRODUCTION

In a previous issue (B. Hutchins, "An Introduction to Adaptive Filters of Several Types", Electronotes, Vol. 16, No. 170, Feb. 1988) we discussed the subject of adaptive filters in some detail. There, adaptive filter ideas were examined using a simulator. Simulators of this type are quite easy to write. The simulator used in the earlier report (having been first developed about 1982) was a bit awkward graphically, however. In fact, what we did was print the output samples by plotting the letter O on paper cycling through the printer. After a good number of photocopy reductions, and cut-and-paste, we were able to present the examples we wanted to show.

The new simulator presented here is probably more convenient, since it runs on BASIC, and presents a display of all four signals of interest (d , e , x , and y) on a single CGA screen. This means that in many (if not most) cases we can demonstrate the basic ideas of an adaptive filter with just a single setup and single screen display.

REVIEW OF ADAPTIVE FILTER BASICS

Recall that the adaptive filter (Fig. 1) has in general two inputs and two outputs. The most familiar input is the so-called "desired" signal $d(n)$, while the most familiar output is the so-called "error" signal $e(n)$. By "most familiar" we mean that these are the two signals out of the four that relate most closely to our usual notions of a one-input, one-output filter. Unfortunately, this means that the terminology "desired" and "error" are apparent misnomers. If the input signal $d(n)$ is really "desired", then why are we taking steps to filter it into some other signal. If the output $e(n)$ is an "error", why is it usually the signal we want to use. However, don't let the terminology mislead you.

Generally the adaptive filter requires a second input which is usually called a "reference" signal, here denoted by $x(n)$. The reference signal is chosen so that it has one or more components that are related (i.e., correlated with) one or more components of the desired signal. We intend to use this reference to help us either cancel or enhance components in the desired (input) signal, before they are output as $e(n)$ or as $y(n)$, respectively.

The second output $y(n)$ is often thought to be internal to the filter, and not a true output. It can of course be a second output, and it is often thought of as an FIR-filtered version of $x(n)$. This view is approximately correct in some cases. However, it is not in

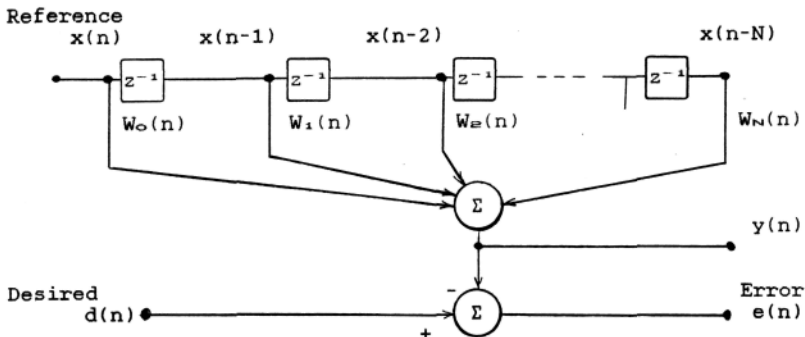


Fig. 1 A Basic Adaptive Filter

general a FIR-filtered version of $x(n)$, because the "tap weights" of the FIR filter are time-varying, at least initially. In fact, $y(n)$ is given by (see Fig. 1 as well):

$$y(n) = \sum_{j=0}^N W_j(n)x(n-j) \quad (1)$$

We can conveniently write $x(n-j)$ as $x_j(n)$ so that $x_j(n)$ is the sample at the j -th tap at time index n , while $W_j(n)$ is the tap weight at the j -th tap at time index n . This means that equation (1) can be rewritten as:

$$y(n) = \sum_{j=0}^N W_j(n)x_j(n) \quad (2)$$

The tap weights $W_j(n)$ are determined for each time n by some calculation involving the signals and parameters. There are a good number of methods for finding $W_j(n)$. For example, the so-called "LMS algorithm" gives the tap weights as:

$$W_j(n+1) = W_j(n) + 2\mu x_j(n)e(n) \quad (3)$$

In practice, μ is usually taken to be a very small number (perhaps 0.01 or 0.001), so in general, $W_j(n+1)$ is little changed from $W_j(n)$ during any single iteration. In order that $W_j(n)$ changes over a significant range, over a good number of iterations of equation (3), it is necessary that $x_j(n)$ and $e(n)$ be correlated over this set of iterations. That is, the small contributions to updating $W_j(n)$ will tend to cancel unless $x_j(n)$ and $e(n)$ are of the same sign over the set of iterations.

FEATURES OF THE SIMULATOR

The code for the simulator is given at the end of this report. It is almost short enough to just type in, although a diskette is available if you wish one.

The program is run, and you are offered a three item menu, the third of which is a return to DOS. To proceed, you will be asked to select either a N=normal setup, or a S=special setup. The special setup allows the user to initialize tap weights, and/or to delay the screen display. If you select S, or if you just choose N, you will be first asked to enter the number of taps. Something like 10 is a good start if you have no better choice yet. You will then be asked to specify the component parts of the reference signal, and then the component parts of the desired signal. For each of the two signals you can choose two sinusoidal components by specifying their amplitude, frequency, and phase. Amplitudes should be 1.0 or less, and frequencies are specified relative to a sampling frequency of 1, so all frequencies will be 0.5 or less. Phase is given in degrees. If you do not need either (or both) sinusoidal components, enter a amplitude of zero. After specifying (or bypassing) the two sinusoidal components, you may specify a random component (noise) with an amplitude of up to about 1.0. Finally, a dc constant component may be chosen (amplitude up to 1.0).

The last entry for the reference is what is called a delay. If you choose any value for the delay (positive) other than 0, your choices above for the reference will be bypassed, and the reference signal will be the desired signal delayed by the number of iterations that you have specified for a delay. This is of course the way we implement a so-called "decorrelating delay" from which a reference signal is effectively derived from the desired signal (recovering a one-input filter). Of course, if you were intending to use the decorrelating delay, you would probably have just entered zeros (or just pressed enter) for the choices above the delay.

After entering both signals, you will need to enter μ , the convergence factor. A good choice is something like 0.01 or 0.001 for a starting test. Following the entry of μ , if you have chosen the special setup, you will be asked to decide if you want to initialize the tap weights (otherwise they will be set to 0). This would normally be used only for a special test - for example, to show that you can get different, but functionally equivalent sets of tap weights for different initial values.

The second special setup test is to choose a delay for the screen display. If you choose 0, there will be no delay. [If you choose N=no for the tap weight initialization, and 0 for the screen delay, you get the equivalent of a N=normal setup, done the hard way.] The purpose of the screen delay is as follows. Normally, you will get a plot of the results for iterations from $n=1$ to $n=240$. This uses up most of the screen. Yet, 240 iterations may not be enough to show a complete convergence to a solution. In such cases, we might want to wait until later iterations before starting the display. For example, if you choose a screen delay of 500, you will get a blank

screen for a period of time, followed by a plot of the results of iterations from 501 to 740.

If during the plotting you see that you have made a setup mistake, you do not have to finish the plot, but can just press e=exit. Following the completion of the plot, you will have the option of running a new test by pressing c=new run, or of looking at the tap weights with w=weights. Of course, and hard copies of the results will depend on your own printer setup, but many printers will give you the printout with just 2nd PrtSc.

EXAMPLES

A classic example of adaptive filtering can be obtained by setting $x(n)$ equal to some sinusoidal, while $d(n)$ contains this same sinusoidal frequency (amplitude and phase may be different) plus some second sinusoidal (or possibly a noise representing an information bearing signal). Try $x=0.6 \cdot \sin(2\pi \cdot 0.05 \cdot n)$ and $d=0.6 \cdot \sin(2\pi \cdot 0.02 \cdot n) + 0.6 \cdot \sin(2\pi \cdot 0.05 \cdot n)$ with a 10-tap filter and $\mu = 0.02$. Fig. 2 shows the results. Note that the component that is correlated between $d(n)$ and $x(n)$, the 0.05 frequency, is cancelled from $e(n)$, but enhanced at $y(n)$. This shows the general notion that correlated components will appear at $y(n)$ while uncorrelated components will appear at $e(n)$. We often think of this as an interference cancellation procedure. That

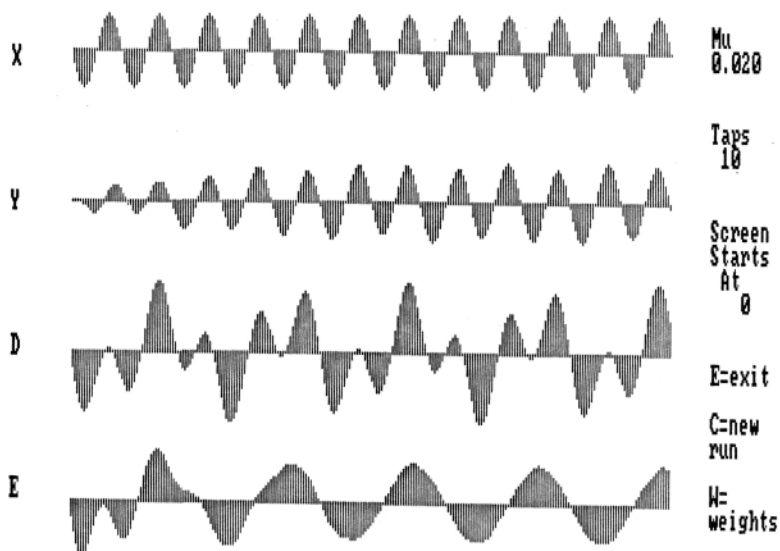


Fig. 2

is, there is a signal with frequency 0.05 that is interfering with (mixed with) the one at frequency 0.02. We obtain a reference to the 0.05 frequency, and apply this to $x(n)$. It is then the purpose of the adaptive linear combiner (the FIR filter structure) to find the correct amplitude and phase to cancel the 0.05 component in $d(n)$, producing $e(n)$.

This idea of $y(n)$ representing correlated components while $e(n)$ corresponds to uncorrelated components can be demonstrated by two extreme examples. Fig. 3 shows a case where both $x(n)$ and $d(n)$ are composed of a single sinusoidal of frequency 0.05. We have chosen these to be 180° out of phase for a more general example. Note that the two signals $x(n)$ and $d(n)$ are perfectly correlated if the correct time shift is achieved. Fig. 3 shows that $e(n)$ (uncorrelated) goes to 0 while $y(n)$ (correlated) becomes $d(n)$. Thus $y(n)$ is the component of $d(n)$ that is correlated with $x(n)$.

The second extreme example is shown by Fig. 4. Here $x(n)$ and $d(n)$ are both random sources, which are uncorrelated. For the most part, $y(n)$ (correlated) remains very small, while $e(n)$ (uncorrelated) closely resembles $d(n)$. The small amount of "accidental" correlation between two otherwise random signals is expected. However, this also tells us (as we suspect) that any old random noise $x(n)$ is not a satisfactory reference signal to cancel random noise at $d(n)$. The noises would have to be correlated.

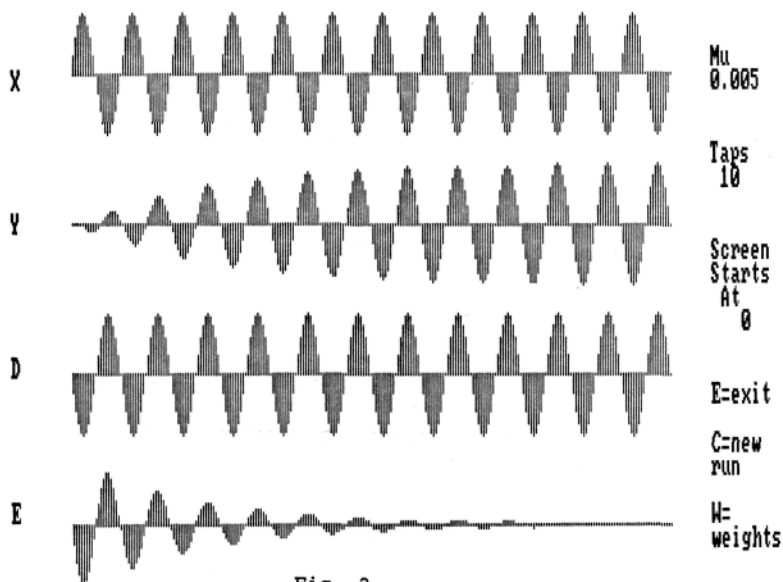


Fig. 3

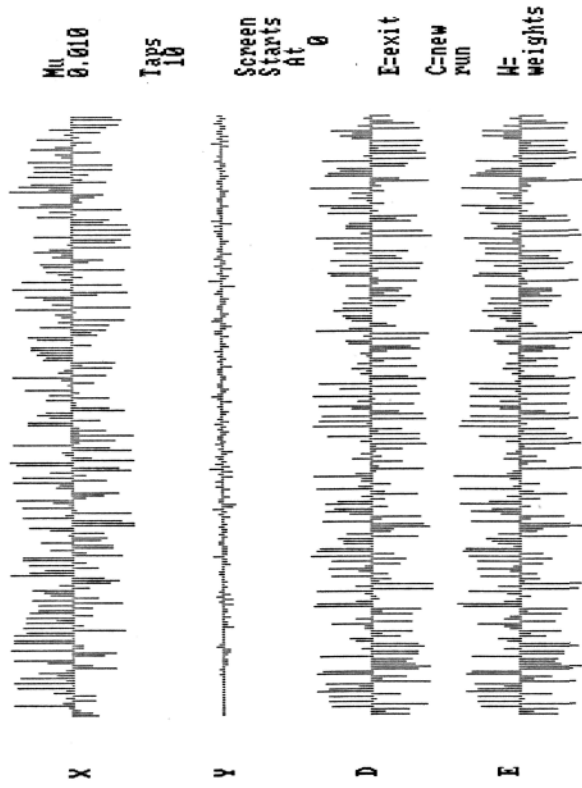


Fig. 4

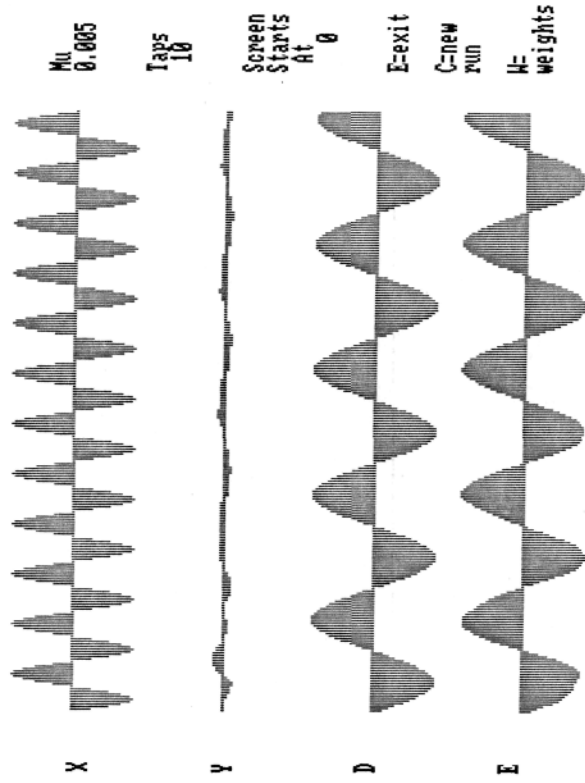


Fig. 5

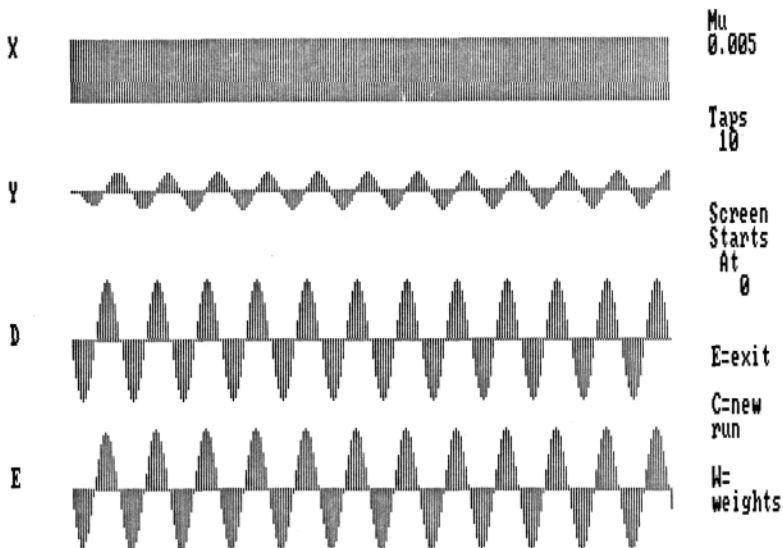


Fig. 6

Fig. 5 is another useful example illustrating the degree of correlation detected. Here $x(n)$ is of frequency 0.05 while $d(n)$ is of frequency 0.02. These signals are basically uncorrelated over long periods of time, but there are small amounts of correlation detected in $y(n)$.

Fig. 6 is another interesting case where we have $x(n)$ set to a constant, while $d(n)$ is of frequency 0.05. If we rely on linear time-invariant filter theory, the output $y(n)$ of the "FIR filter" which has input $x(n)$ could be nothing other than a constant (after any transients, which would be of finite duration in this case). Yet we find an output that is sinusoidal, of frequency 0.05. This is due to the time varying nature of the tap weights. Here, the error $e(n)$ is clearly not zero, but the taps continually try to reduce the error. Another way to look at it is that the taps are trying to convert 0 frequency to a frequency of 0.05.

A related example (not illustrated) can be found by making $x(n)$ a frequency of 0.04 for example, and $d(n)$ a frequency of 0.05. The tap weights will shift in time, with a pattern moving from right to left, trying to convert the 0.04 frequency to a 0.05 frequency in a manner that reminds us of a Doppler shift.

Fig. 7 shows an example of great importance. Here there is no established $x(n)$, but rather $x(n)$ is a delayed (by 50 iterations) version of $d(n)$, as can be easily seen. The signal $d(n)$ is composed of a sinusoidal of frequency 0.05 plus an equal amplitude of noise. What we see is first of all a separation of signals. The signal $y(n)$

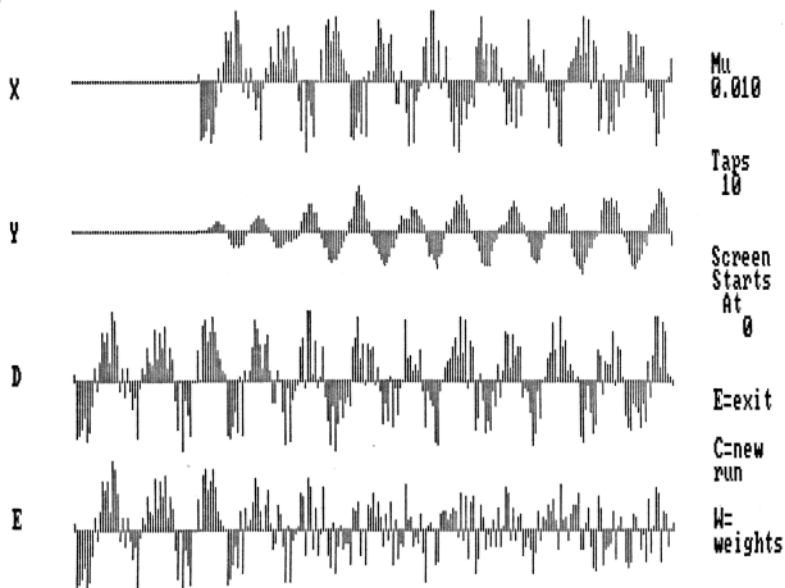


Fig. 7

takes on the sinusoidal component, while $e(n)$ takes on the random component. Thus we look at this as a signal separation example. It is common (and sometimes misleading) to look at this type of example from an application point of view. In one application, $d(n)$ is a noisy sinusoidal waveform to be cleaned up (i.e., $y(n)$ is the output). In another example, $d(n)$ is an information bearing signal (speech as represented by a random sequence for example) that has been corrupted by a sinusoidal (such as the pick-up of AC "hum") in which case, $e(n)$ is the cleaned output.

The example of Fig. 7 is of great importance because it is a one input filter. We do not have to get a separate reference signal, and in many cases, it is difficult to see how we could expect to have one available. Here the delay between $d(n)$ and $x(n)$ is called a "decorrelating delay". The sinusoidal component in $d(n)$ is correlated over a length exceeding the 50 delays, while the noise component is not correlated over 50 delays. [In fact, a delay of 1 would have worked for this illustration!] In this case, $y(n)$ represents the correlated components of $d(n)$ while $e(n)$ represents the uncorrelated components of $d(n)$. Note that because the reference signal is not "pure", some noise appears in the $y(n)$ output while some of the sinusoidal shape seems to be still present in $e(n)$.

There are many more interesting adaptive filter examples, but here we are illustrating the simulator, and only making a few points about adaptive filtering. Accordingly we will skip to two final examples, shown in Fig. 8a and Fig. 8b, which will illustrate the use of the screen delay feature. Here both the desired signal $d(n)$ and

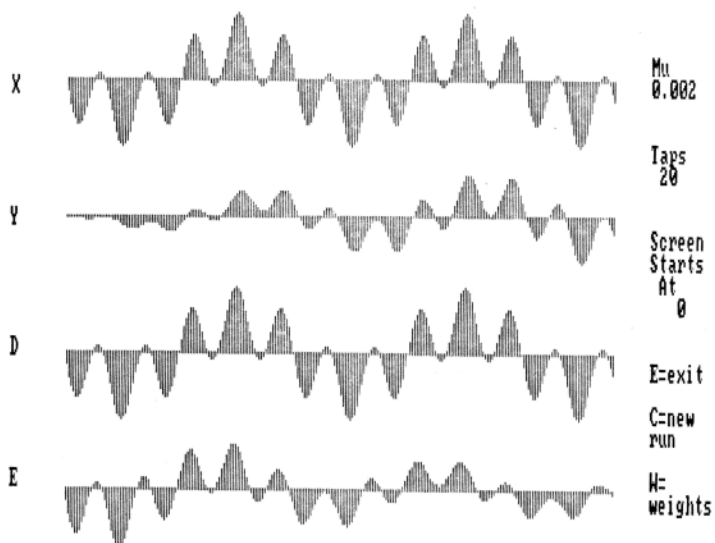


Fig. 8a

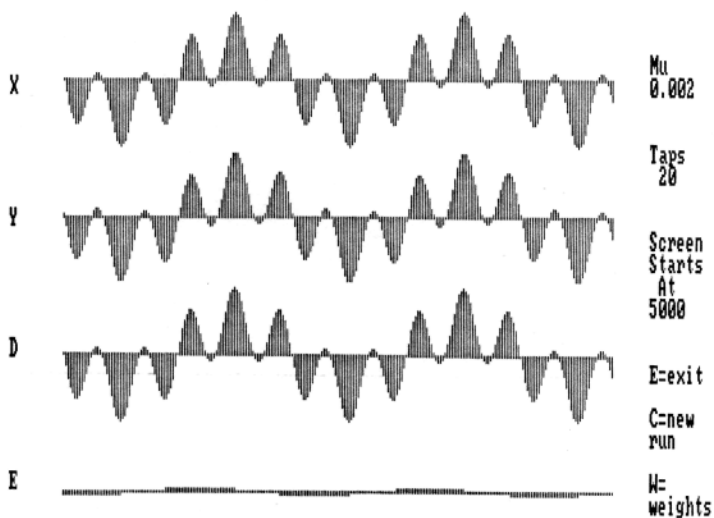


Fig. 8b

the reference signal are composed of frequencies 0.01 and 0.05 at amplitudes 0.6. That is, they are identical, and we might well expect a complete cancellation (similar to Fig. 3). However, we have chosen a length of 20 taps and a μ of 0.002. In consequence, it is difficult to be certain how much cancellation will eventually result, based on the first 240 iterations of Fig. 8a. This is where the screen delay is useful. Fig. 8b shows the same setup after 5000 iterations. We note that $e(n)$ is very small relative to $d(n)$, but has not become zero (at least, not yet).

THE PROGRAM CODE

```

10 REM save "AD"
20 KEY OFF
30 TP=2*3.14159
40 DIM W(50),X(50),DS(100)
50 CLS:SS=0:SCREEN 0:IDS=0
60 FOR M=0 TO 50:X(M)=0:W(M)=0:DS(M)=0:NEXT
70 E=0:D=0:Y=0
80 COLOR 14
90 PRINT:PRINT "                ADAPTIVE FILTER SIMULATOR"
100 PRINT:COLOR 15
110 PRINT:PRINT "                N = Normal Setup
                        S = Special Setup
                        D = Return to DOS"

120 Z$=INKEY$
130 IF Z$="n" OR Z$="N" THEN 160
140 IF Z$="d" OR Z$="D" THEN SYSTEM
150 IF Z$="s" OR Z$="S" THEN SS=1:GOTO 160 ELSE 120
160 COLOR 11
170 INPUT "                Number of Taps:      ",NT
180 PRINT
190 INPUT "                Reference Amp #1:        ",RA1
200 INPUT "                Reference Freq #1:       ",RF1
210 INPUT "                Reference Phase #1:      ",RP1
220 INPUT "                Reference Amp #2:       ",RA2
230 INPUT "                Reference Freq #2:      ",RF2
240 INPUT "                Reference Phase #2:     ",RP2
250 INPUT "                Reference Noise:        ",RN
260 INPUT "                Reference DC:           ",RDC
270 INPUT "                Reference Delay:        ",RD
280 PRINT
290 INPUT "                Desired Amp #1:           ",DA1
300 INPUT "                Desired Freq #1:       ",DF1
310 INPUT "                Desired Phase #1:      ",DP1
320 INPUT "                Desired Amp #2:       ",DA2
330 INPUT "                Desired Freq #2:      ",DF2
340 INPUT "                Desired Phase #2:     ",DP2
350 INPUT "                Desired Noise:        ",DN
360 INPUT "                Desired DC:           ",DDC
370 PRINT
380 INPUT "                Mu:                       ",MU
390 PRINT

```

```

400 IF SS=1 THEN 410 ELSE 450
410 PRINT "      Initialize Taps to Other Than 0? (Y,N)  "
420 Z$=INKEY$
430 IF Z$="Y" OR Z$="y" THEN 970
440 IF Z$="N" OR Z$="n" THEN 445 ELSE 420
445 INPUT "      Number of Iterations to Delay Screen Display  ";IDS
450 REM      MAIN ALGORITHM
451 CLS
460 SCREEN 2
470 LOCATE 4,2:PRINT "X"
480 LOCATE 4,72:PRINT USING "#.###";MU
490 LOCATE 3,72:PRINT "Mu"
500 LOCATE 8,72:PRINT USING "###";NT
510 LOCATE 7,72:PRINT "Taps"
511 LOCATE 11,72:PRINT "Screen"
512 LOCATE 12,72:PRINT "Starts"
513 LOCATE 13,72:PRINT " At "
514 LOCATE 14,72:PRINT USING "#####";IDS
520 LOCATE 10,2:PRINT "Y"
530 LOCATE 16,2:PRINT "D"
540 LOCATE 22,2:PRINT "E"
541 LOCATE 17,72:PRINT "E=exit"
550 FOR T=1 TO 240+IDS
560 FOR M=NT TO 2 STEP -1
570 X(M)=X(M-1)
580 NEXT
590 IF RD=0 THEN 680
600 FOR M=100 TO 2 STEP -1
610 DS(M)=DS(M-1)
620 NEXT
630 DS(1)=D
640 IF RD>0 THEN 650 ELSE 680
650 IF T>RD THEN 660 ELSE 670
660 X(1)=DS(RD):GOTO 690
670 X(1)=0:GOTO 690
680 X(1) = RA1*SIN(TP*RF1*T + RP1/57.29578) + RA2*SIN(TP*RF2*T +
      RP2/57.29578) + RN*(2*RND - 1) + RDC
690 Y=0
700 FOR M=NT TO 2 STEP -1
710 Y = Y+X(M)*W(M)
720 NEXT
730 D = DA1*SIN(TP*DF1*T + DP1/57.29578) + DA2*SIN(TP*DF2*T +
      DP2/57.29578) + DN*(2*RND - 1) + DDC
740 E=D-Y
750 FOR M = NT TO 1 STEP -1
760 W(M)=W(M)+2*MU*X(M)*E
770 NEXT
775 IF T<IDS THEN 830
780 S=2*(T-IDS)+55
790 LINE (S,25)-(S,(25+20*X(1)))
800 LINE (S,75)-(S,(75+20*Y))
810 LINE (S,125)-(S,(125+20*D))
820 LINE (S,175)-(S,(175+20*E))
830 Z$=INKEY$:IF Z$="e" OR Z$="E" THEN 50
840 NEXT

```

```

841 LOCATE 19,72:PRINT "C=new"
842 LOCATE 20,72:PRINT "run"
843 LOCATE 22,72:PRINT "W= "
844 LOCATE 23,72:PRINT "weights"
850 Z$=INKEY$
860 IF Z$="W" OR Z$="w" THEN 890
870 IF Z$="c" OR Z$="C" THEN 880 ELSE 850
880 SCREEN 0:GOTO 50
890 CLS
900 SPACE=INT(500/NT)
910 LINE (0,75)-(600,75)
920 FOR M=1 TO NT
930   LINE (M*SPACE,75+100*W(M))-(M*SPACE,75)
940 NEXT
950 LOCATE 2,25:PRINT "      TAP WEIGHTS"
960 GOTO 850
970 FOR M=1 TO NT
980 PRINT "      Tap No ";M;:INPUT W(M)
990 NEXT
1000 GOTO 450

```

* * * * *

Two Roads to Narrow-Band Chaos

(Continued from Page 2)

non-linear to a significant degree, at least until this possibility is ruled out. Further, and perhaps more importantly, there is the interaction of the processing mechanism with the excitation mechanism which may have a highly non-linear effect. For example, it might be the case that a relatively small pressure variation in an air column, due to a standing wave resonance, is enough to cause a reed to open fully, and suddenly, from a closed position.

It is useful to make sure we do not get carried away with chaos in musical-tone synthesis, or in musical composition, or in any application, for that matter. That chaos occurs naturally in our experiences is certain. It is also certain that many of us didn't have a name for what we saw, until very recently, when we learned of the science of chaos. Yet this does not mean that chaos is important in all cases. A cup of water filled by a regular drip, a chaotic drip, or a steady stream drinks exactly the same.

Ultimately we must address questions of practicality and implementation. It is not enough that narrow-band chaos might be useful for tone synthesis. Can we do it fast enough? Can we adjust the pitch to needed values? Here is where driven systems are particularly attractive. That is, can we use a driven chaotic system much as we have used conventional animators of the past? In such a case, the input signal itself determines the pitch.

* * * * *

ELECTRONOTES, Vol. 16, No. 178, July 1991
 Published by B. Hutchins, 1 Pheasant Lane, Ithaca, NY 14850
 Phone 607-273-8030