ON A MICROCOMPUTER IMPLEMENTATION OF AN INTRUSION-DETECTION ALGORITHM

by

KENNETH JOSEPH HASS

B. S., Kansas State University, 1978

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1979

Approved by:

Donald W. Lechat

Major Professor

This work is dedicated to my parents, for giving me twenty-two years of devotion and friendship.

THIS BOOK CONTAINS NUMEROUS PAGES WITH DIAGRAMS THAT ARE CROOKED COMPARED TO THE REST OF THE INFORMATION ON THE PAGE.

THIS IS AS RECEIVED FROM CUSTOMER.

TABLE OF CONTENTS

I. Introduction

The notion of using an ADP to improve the performance of perimeter sensors for intrusion-detection was introduced in [1]. Such sensors are buried cables along the perimeter of an area to be protected. The response (output) of a cable due to a signal source depends upon the type of source and its proximity to the source.

Perimeter sensors respond not only to intruder stimuli, but also to a variety of other stimuli (e.g., noise sources), resulting in poor signal-to-noise ratio. Intruder stimuli in this case are transient signals, as exemplified in Fig. 1. The noise present may be periodic or random, or a combination of both. Since the random noise encountered is non-stationary, or at best stationary on a short-term basis, an adaptive approach involving an ADP is employed. The basic idea is to use the ADP to remove the correlated portion of the noise input. As such, it assists in reducing false (nuisance) alarms in the absence of intruder stimuli. On the other hand, it yields an improvement in signal-to-noise ratio when intruder stimuli are present, thereby assisting in their detection.

There are several algorithms that are available for implementing the ADP--e.g., see [1-6]. However, this research was restricted to a modified version of Widrow's LMS (least-mean-square) algorithm [2], which shall be referred to as the MLMS algorithm [7].

The intrusion-detection scheme considered consists of the ADP in cascade with an ATD algorithm; see Fig. 2. The ATD algorithm is "adaptive"

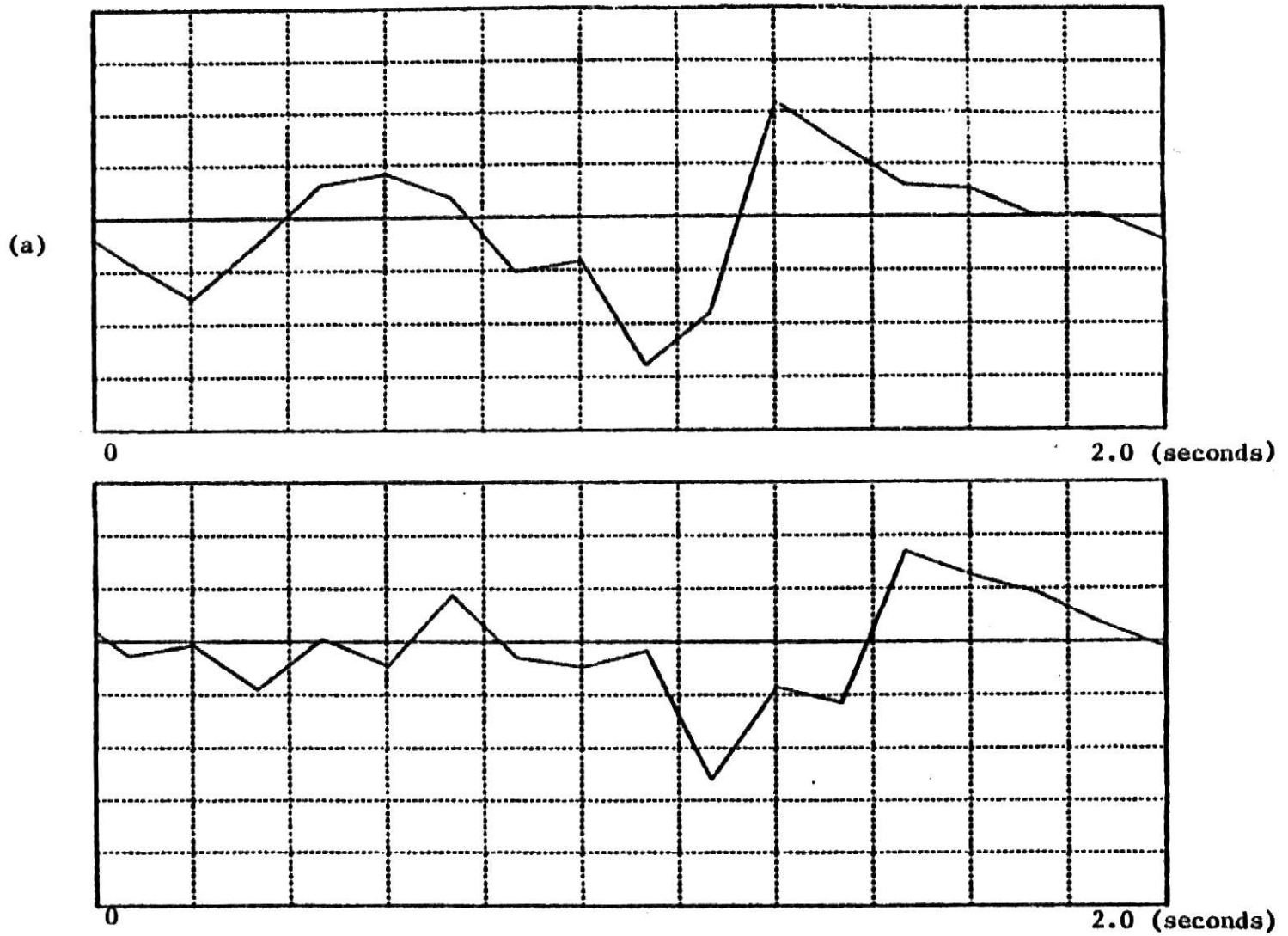

Fig. 1. Two examples of intruder stimuli caused by an intruder (walker) crossing the sensor (cable) once; sampling frequency = 8 sps.

in the sense that it involves a pair of variances which are continually updated as the ADP output is processed. A certain function (say d_m) of this pair of variances is computed at time m, and them compared to a preselected threshold θ. An intrusion is indicated by an alarm condition at the output of the ATD stage whenever $d_m \geq \theta$.

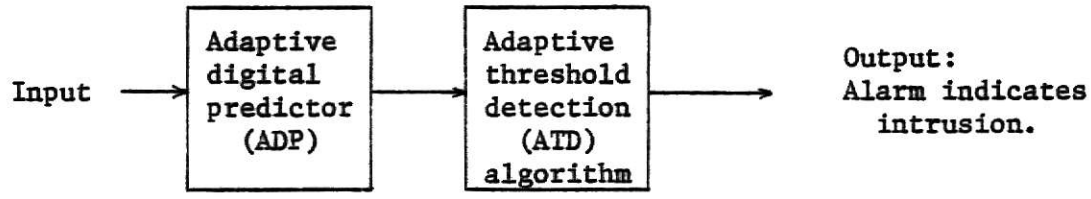

Fig. 2. Block diagram of intrusion-detection scheme.

The main objective of this paper is to discuss various aspects of obtaining a fixed-point implementation of the configuration in Fig. 2, using a microcomputer system [8]. In particular, this work involved a TM 990/100M microcomputer system, which includes the following: a TMS 9900 microprocessor; 256 16-bit words of RAM (random access memory); 2048 words of EPROM (erasable programmable read only memory); a TMS 9901 programmable system interface; a TMS 9902 asynchronous communications controller; and, the pertinent control logic. Its unique features include a hardware multiply/divide capability and a "workspace" register. The processor operates at 3 MHz, thus requiring 4.67 and 17.3 microseconds for a register-to-register move, and a multiplication, respectively. The implementation of the overall intrusion-detection scheme is explained in a step-by-step manner via a flowchart representation. Some experimental results involving data acquired via field experiments are also included.

II. The ADP

The ADP configuration is shown in Fig. 3, where the delay parameter Δ is fixed and equals 1. If g_m denotes the ADP output at time m, it follows that

$$g_m = \sum_{n=1}^{N} b_{n,m} f_{m-n} \tag{1}$$

where

N is the number of predictor coefficients (weights),

$b_{n,m}$ is the n-th weight at time m, and

f_m is the input sample at time m.

In Fig. 3, the MLMS algorithm is used to update the weights $b_{n,m}$ as follows:

$$b_{n,m+1} = (1 - u)\, b_{n,m} + v f_{m-n} \tag{2}$$

where

$0 < u < 1$ is a scale factor, and

v is the convergence parameter.

From (2) it follows that the special case u = 0 results in the conventional LMS algorithm [2]. The reason for introducing the term (1 - u) will be apparent later.

It can be shown that the Wiener solution B_{opt} associated with the expected value of $b_{n,m}$ in (2) is given by

$$B_{opt} = [uI + R_{ff}]^{-1} P_{yf} \tag{3}$$

where

R_{ff} is the input autocorrelation matrix

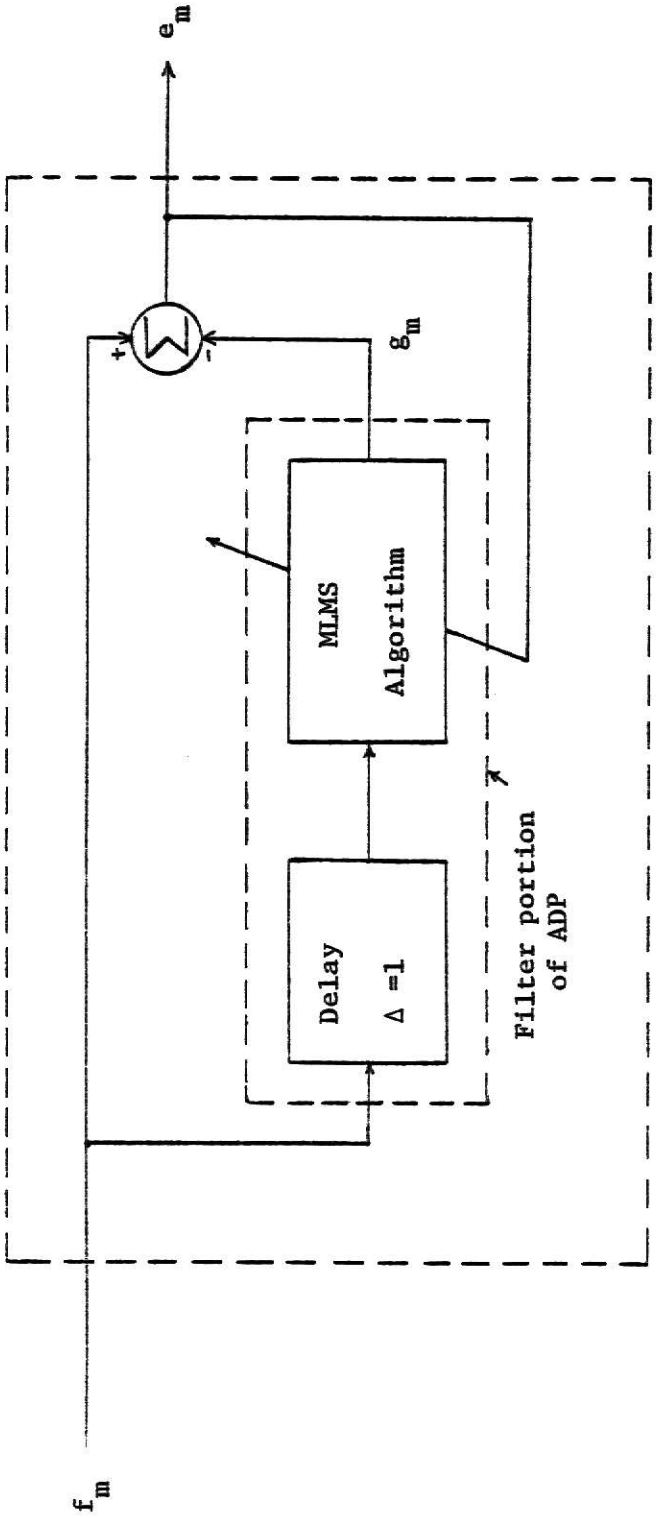

Fig. 3. Adaptive Digital Predictor Configuration

I is the identity matrix, and

P_{yf} is a crosscorrelation vector whose elements are given by

$$E\{f_m f_{m-i}\} , \quad i=1,2,\ldots,N .$$

Note that inclusion of the term (1 - u) in (2) is equivalent to adding a trace of white noise to the input data. This is because I represents the correlation matrix of white noise. As such, the exact Weiner solution $B^* = R_{ff}^{-1} P_{yf}$ is not attained but one can come arbitrarily close to it by adjusting the parameter u in (2).

The reason for introducing the term (1 - u) in (2) is that it avoids an undesirable long-term effect, which has been referred to as a "no-pass" phenomenon. Basically this implies that over a period of time, an ADP (with sufficient number of weights) will not only remove noise, but intruder stimuli as well. In other words, it tends to become a "no-pass" filter. An analysis of this phenomenon is available elsewhere [7,9], and hence will not be discussed here. It can be summarized it as follows:

> The ADP first adapts to higher levels in the input, decorrelating as much as it can within its own limits in size. It then adapts to low-level components. To this end, the ADP creates an overall transfer function of zero for these components.
>
> Thus, the filter portion of the ADP (see Fig. 3) evolves until its amplitude response is near unity for all the frequency components present in the input, <u>regardless</u> of their size. In general, a predictor of finite length cannot eliminate all signals present, but its gain and phase shift will be sufficient across the spectrum to remove most of the correlated input components. Hence, the filter

portion achieves an "all-pass" mode of operation, giving the overall ADP the appearance of a "no-pass" filter.

To examine the behavior of the MLMS algorithm, assume that for some $m \geq M$, the error term in (2) is negligible; i.e., at time $m = M$, the ADP weights have converged to B_{opt} in (3). Then, (2) simplifies to yield:

$$\begin{aligned} b_{n,M+1} &= (1 - u)\, b_{n,M} = (1 - u)\, b_{n,opt} \\ b_{n,M+2} &= (1 - u)^2\, b_{n,opt} \\ &\vdots \\ b_{n,M+k} &= (1 - u)^k\, b_{n,opt} \quad . \end{aligned} \tag{4}$$

From (4) it is apparent that $b_{n,M+k}$ tends towards zero with increasing k, since $(1 - u) < 1$. This in turn causes the error term e_m in (2) to increase, hence causing the ADP to readapt. The adaptation process causes b_n to move back toward $b_{n,opt}$ until e_m is negligible; then the entire foregoing cycle repeats. It is this feature that avoids the no-pass phenomenon, since the ADP is constrained to focus its attention on decorrelating only the higher levels of the input.

III. Detection Considerations

Detection at the output of the ADP can be attempted via a variety of classical techniques, an excellent discussion of which is available in [10,11]. In this study, the ATD algorithm was considered as illustrated in Fig. 2. A brief derivation of the same follows.

Begin by making the following assumptions:

(1) Successive noise and intruder samples at the ADP output have Gaussian distributions with zero means and variances σ_n^2 and σ_s^2, respectively.

(2) Noise and intruder samples are uncorrelated.

From these assumptions, it follows that

$$f(q_j|0) = \frac{1}{\sqrt{2\pi}\,\sigma_n} e^{-q_j^2/2\sigma_n^2}$$

and

$$f(q_j|s) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-q_j^2/2\sigma^2}, \quad |q_j| < \infty \tag{5}$$

where

q_j is the ADP output sample at time j,

$f(q_j|0)$ is the conditional probability density function, given that no intruder is present--i.e., $q_j = n_j$,

$\sigma^2 = \sigma_s^2 + \sigma_n^2$,

and

$f(q_j|s)$ is the conditional probability density function, given that an intruder is present--i.e., $q_j = n_j + s_j$.

Now, using a likelihood ratio approach [10], the decision rule is that an intruder is present if

$$\sum_{j=1}^{M} \ell n \left\{\frac{f(q_j|s)}{f(q_j|0)}\right\} \geq K_1 \tag{6}$$

where K_1 is a constant.

Substitution of (5) in (6) leads to

$$M \ell n \left\{\frac{\sigma_n}{\sigma}\right\} + \frac{1}{2}\left(\frac{1}{\sigma_n^2} - \frac{1}{\sigma^2}\right) \sum_{j=1}^{M} q_j^2 \geq K_1$$

which implies that it can be decided an intruder is present if

$$\sum_{j=1}^{M} q_j^2 \geq \frac{2}{\left(\frac{1}{\sigma_n^2} - \frac{1}{\sigma^2}\right)} \left[K_1 + \frac{M \ell n\left\{\frac{\sigma^2}{\sigma_n^2}\right\}}{2}\right] .$$

Since $\sigma^2 = \sigma_s^2 + \sigma_n^2$, the above equality can be expressed as

$$\sum_{j=1}^{M} q_j^2 \geq 2\sigma_n^2 \left(1 + \frac{\sigma_n^2}{\sigma_s^2}\right)\left[K_1 + \frac{M}{2} \ell n\left\{1 + \frac{\sigma_s^2}{\sigma_n^2}\right\}\right] .$$

Note that the term σ_s^2/σ_n^2 is the signal-to-noise ratio at the output of the ADP. Thus, if it is assumed that σ_s^2/σ_n^2 is appreciably larger than 1, it follows that

$$\sum_{j=1}^{M} q_j^2 \geq 2\sigma_n^2\left[K_1 + \frac{M}{2} \ell n\left\{\frac{\sigma_s^2}{\sigma_n^2}\right\}\right] .$$

or,

$$\frac{1}{M} \sum_{j=1}^{M} q_j^2 \geq K_2 \sigma_n^2 \tag{7}$$

where

$$K_2 = 2\left[\frac{K_1}{M} + \frac{1}{2} \ell n\left\{\frac{\sigma_s^2}{\sigma_n^2}\right\}\right] .$$

From (7) it can be concluded that for stationary inputs, the optimum rule is to decide that an intruder is present if the variance of the ADP output is greater than or equal to a quantity which is proportional to the variance of the noise process. In practice, however, the noise is stationary only on a short-term basis. Further, not all of the assumptions made in connection with deriving the above decision rule may be valid. Thus, the decision rule is modified to obtain the following suboptimum one, which states that an intruder is present if

$$\sigma_j^2 \geq K\, \sigma_{n,j-D}^2 + \theta \tag{8}$$

where

K and θ are constants

σ_j^2 is the variance estimate of the ADP output at time j,

and

$\sigma_{n,j-D}^2$ denotes the corresponding noise variance which is estimated D samples earlier than time j.

In (8), σ_j^2 is estimated at time j using M previous samples of the ADP output, while σ_{j-D}^2 is continually estimated using L samples, as illustrate in Fig. 4. The delay term D is introduced to insure sufficient time lag between the current output (which may include an intruder component), and a noise segment which is used to estimate the corresponding σ_{j-D}^2.

It is apparent that (8) can be rewritten as

$$\sigma_j^2 - K\, \sigma_{n,j-D}^2 \geq \theta \quad . \tag{9}$$

which shall be referred to as the ATD algorithm. It states that whenever the difference signal $(\sigma_j^2 - K\, \sigma_{n,j-D}^2)$ is greater than or equal to a threshold θ, we decide that an intruder is present. Such a decision results in an alarm condition, as illustrated in Fig. 2.

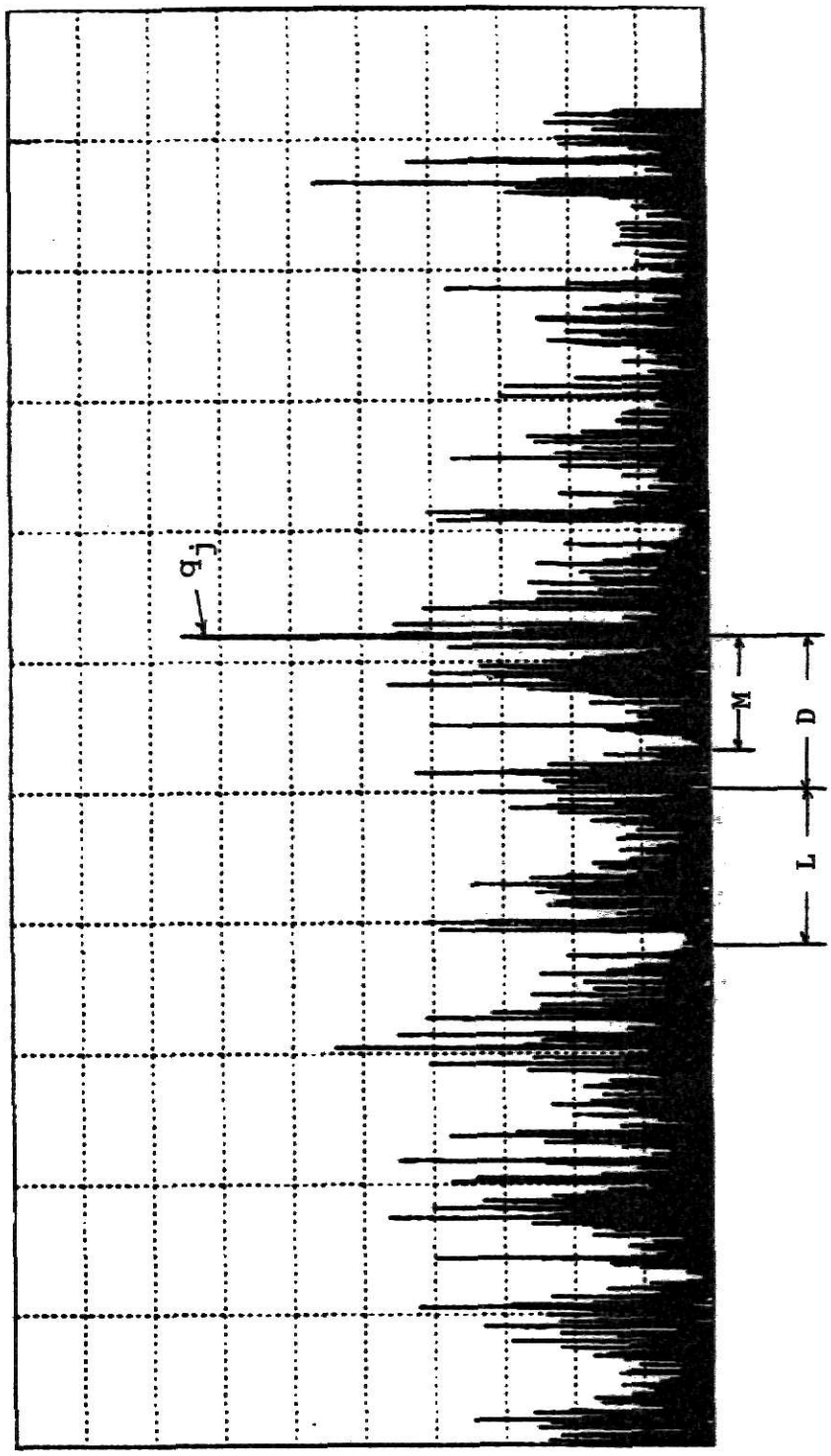

Fig. 4. Pertaining to the ATD algorithm

IV. The TM 990/100M Microcomputer System

A Texas Instruments TMS 9900 microprocessor is the central processing element of this system. It is an NMOS microprocessor with a 16 bit word length. A versatile memory-to-memory architecture is used which allows the user to create multiple register files in memory. These registers are addressed in a manner which is similar to addressing the accumulators of more conventional processors. As a result, the programmer can access 16 registers with high speed single word instructions at any desired time. The programmer is allowed to create or use another bank of registers by performing a two-word instruction. There are only three internal registers accessible to the user: the program counter, which contains the address of the next instruction to be executed; the workspace pointer, which specifies the first address of the register file currently in use; and a status register.

The instruction set includes very powerful data transfer instructions and arithmetic operations. Some instructions provide as many as 16 distinct addressing modes. The single instruction multiply/divide operations are the most unique aspect of the instruction set and have proven to be very valuable in this application. The TMS 9900 executes an unsigned multiply with a 32 bit product in 52 clock cycles (i.e., 17.3 microseconds).

There are 16 priority levels available for external interrupts, plus a non-maskable level for the reset function. When an interrupt occurs, the TMS 9900 loads the workspace pointer and program counter from specified memory locations and begins execution with the new program counter. The

interrupt mask is then set to allow only interrupts of a higher priority than the one being serviced. The interrupt mask can be set to any value using a two word instruction.

A. Programmable Systems Interface

Normal input and output functions are provided by a TMS 9901 programmable system interface. The I/O structure of the TMS 9900-9901 combination permits it to address 4096 individual bits or concatenate up to 16 bits anywhere within that field. Individual bits can be tested (input), set, or reset with single word instructions. Words of 2 to 16 bits can be output or input in parallel with a single word instruction. The TMS 9901 also provides a programmable real-time clock which interrupts the processor at user defined intervals.

B. System Memory

The TM 990/100M microcomputer is supplied with 256 words (expandable to 512 words) of read/write memory, half of which is reserved for monitor workspaces and interrupt vectors. It also contains a versatile monitor, TIBUG, and a line-by-line assembler on EPROM. These blocks of memory are supplemented by a TM 990/201 memory expansion board. This board is populated with 2048 words of RAM and 8192 words of EPROM. An EPROM programmer has been built and the supporting software was written. Several programs are available which allow the user to program, test, or read the contents of an EPROM in the programmer socket.

C. Analog Interface

Analog I/O is provided by an Analogic ANDS 3001 data acquisition system module, which is a single board module that is compatible with

the TMS 9900 system bus. It consists of a 12-bit analog-to-digital (A/D) converter with a conversion rate of up to approximately 30 kHz, and a 12-bit digital-to-analog (D/A) converter. This interface allows for several optional modes of operation. For the current application, it is configured for two's-complement digital input/output, and the maximum analog input/output was set at $\pm$ 10.24 volts.

D. Parallel Digital Interface

A 16-bit bidirectional I/O port was memory mapped into the system on a TMS 990/512 prototyping board. The port consists of four 74C373 octal latches with tri-state outputs. Sixteen of the latches serve as a latched output port and sixteen are used as tri-state input buffers, with the outputs following the corresponding inputs. A 74LS138 decoder is used to provide address selection and separate read/write signals. The port is configured so that it responds to the same addresses as the ANDS 3001 analog interface. This port was constructed so that the TMS 9901 could be reserved for handshaking with a NOVA minicomputer while transferring data in either digital or analog form.

E. Minicomputer Handshaking

Data used for laboratory testing of the algorithm was supplied by a NOVA minicomputer with a Data General Data Acquisition and Control Subsystem (DGDAC). The data files are stored on a hard disk and are transferred under NOVA program control. Selected outputs from the algorithm are simultaneously read into the minicomputer and stored on disk for future analysis; see Fig. 5.

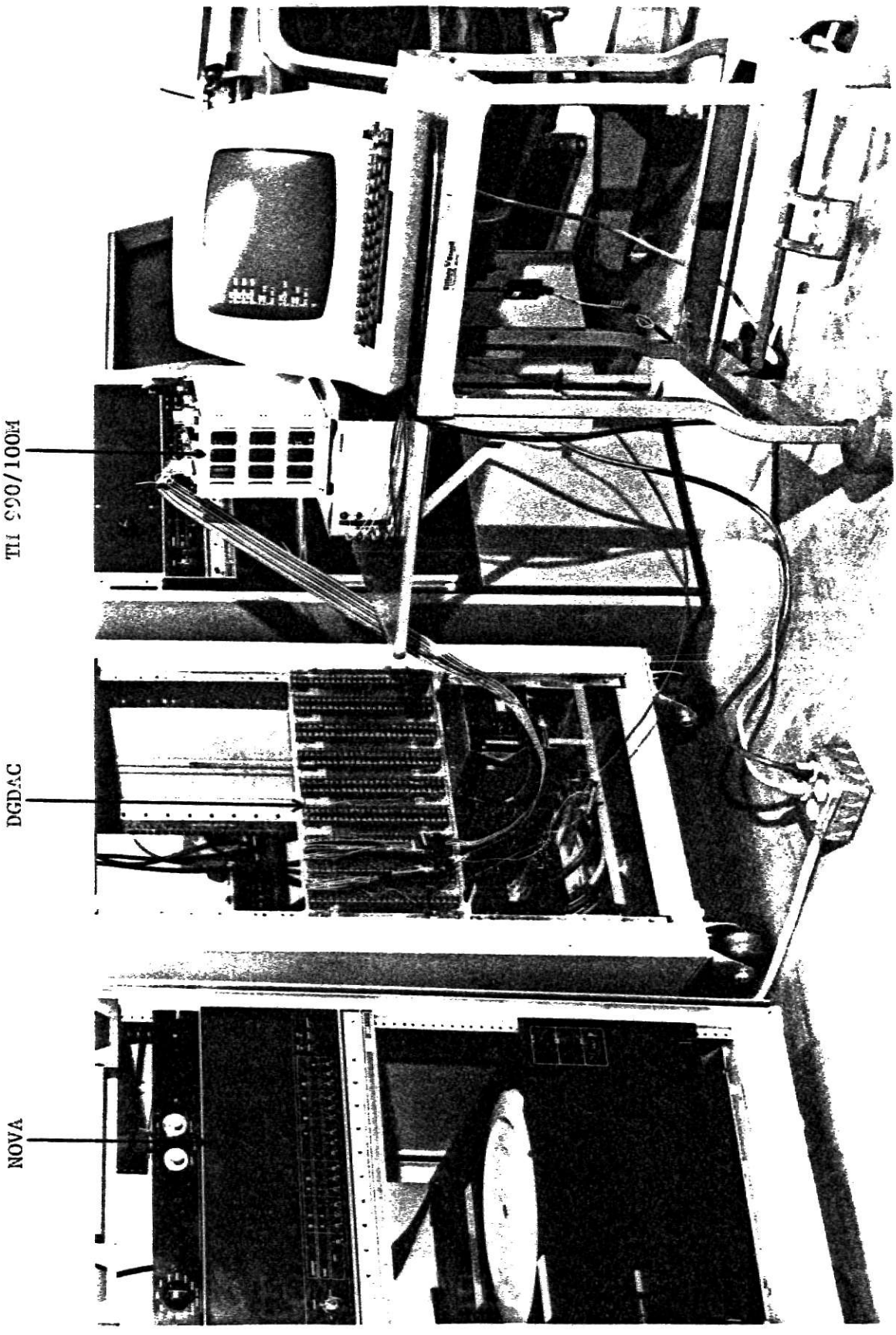

Fig. 5. Configuration for laboratory testing

Handshaking between the NOVA and the TM 990 system requires two control lines. When the NOVA has a data word ready for transfer, an unused D/A output is set to 5 volts. The TMS 9900 senses this as a 1 on a TMS 9901 input bit, reads the data word, and signals its acceptance by forcing a TMS 9901 output bit low for 3 μs. This signal is connected to an external interrupt line of the DGDAC. When the NOVA senses this interrupt, it lowers the D/A output. Both computers process independently from this point, until the TMS 9900 has completed its operations and is ready to output a data word. After the output word is made available to the NOVA, the TMS 9900 again strobes the DGDAC external interrupt line. It then returns to the beginning of the algorithm and waits for a ready level from the NOVA. When the NOVA is ready to input a data word, it waits for the interrupt line to be strobed and latches the data in. The next data word is sent to the DGDAC output and this process is repeated until the test file is exhausted.

V. Fixed Point Notation

The objective in Section VI will be to discuss the overall algorithm in terms of its fixed point implementation. To this end, a convenient notation to represent fixed-point numbers [12] is introduced.

Every constant and variable will be represented in the form (S/I/F), where: S is the number of sign bits, I is the number of data (integer) bits to the left of the binary point, and F is the number of data (fraction) bits to the right of the binary point. Note that the maximum numerical value a binary number can assume is less than 2^I.

A. Addition/Subtraction

There are four rules associated with two numbers that are to be added or subtracted, so that no overflow will occur. These are as follows:

1) The two numbers must have the same format, i.e., they must have the same location for the binary point. In terms of the (S/I/F) notation, the quantity S + I must be equal for the two numbers.

2) Each number must have at least two sign bits, i.e., $S \geq 2$. This prevents a carry out of the data bits from destroying the sign bit.

3) The number of sign bits in the result must be one less than the minimum number of sign bits in the two numbers being added or subtracted.

4) The number of integer bits (to the left of the binary point) in the result must be one more than the maximum number of integer bits in either of the operands.

When adding lists of N numbers, the above rules are modified as follows:

1) Each number must have at least $(\log_2 N + 1)$ sign bits.

2) The number of sign bits in the sum must be equal to the minimum number of sign bits in any of the addends, minus $\log_2 N$.

3) The number of integer bits in the sum must be equal to the maximum number of integer bits in any of the addends, plus $\log_2 N$.

B. Multiplication

When multiplying fixed point numbers, the corresponding rules are much simpler, and the related operands need not have the same format. They are as follows:

1) The number of sign bits in the product must be equal to the sum of the number of sign bits in the operands.

2) The number of integer bits in the product must be equal to the sum of the number of integer bits in the operands.

3) The number of significant fraction bits in the product must be equal to the sum of the number of fraction bits in the operands.

VI. Fixed Point Implementation Considerations

A detailed discussion pertaining to implementing the MLMS and ATD algorithms on the microcomputer system using fixed point arithmetic is now presented. For convenience, consider an ADP with 16 weights, and with parameters $v = 0.125$, and $\hat{u} = 1 - u = 0.875$; see (2). Again, the parameters related to the ATD algorithm will be as follows: $L = 64$, $K = 1$, $D = 16$, $M = 16$, and $\theta = 0.016846$; see Fig. 4 and (9). All input data values f_m to the ADP (see Fig. 3) will be assumed to be fractional numbers.

As described in Section IV.E., the input data to the microcomputer is obtained from a NOVA minicomputer system; see Fig. 5. In block 1 of the flowchart in Fig. 6, an input sample f_m is read in as a two's-complement 12-bit number which is left-justified in the TMS 9900 16-bit data word. Data is transferred in this fashion so that the results obtained will be comparable to those obtained using the ANDS 3001 analog interface. The input word is then shifted right one bit to fix the format of f_m to (2/0/11). Thus, the maximum absolute magnitude of f_m is slightly less than unity. After the receipt of f_m is acknowledged (block 3, in Fig. 6), the ADP routine is executed in block 4. The format of each b_k is set to (1/0/15), so that the result of each multiplication $f_{m-k} * b_k$ has the format (3/0/13).[†] Since there are 16 products to be added, five sign bits are needed in each product to ensure that no overflow will occur. However, for the type of data encountered, g_m was

---

[†]A multiplication results in a 32 bit product which we truncate to 16 bits. This procedure is carred out on all multiplications encountered.

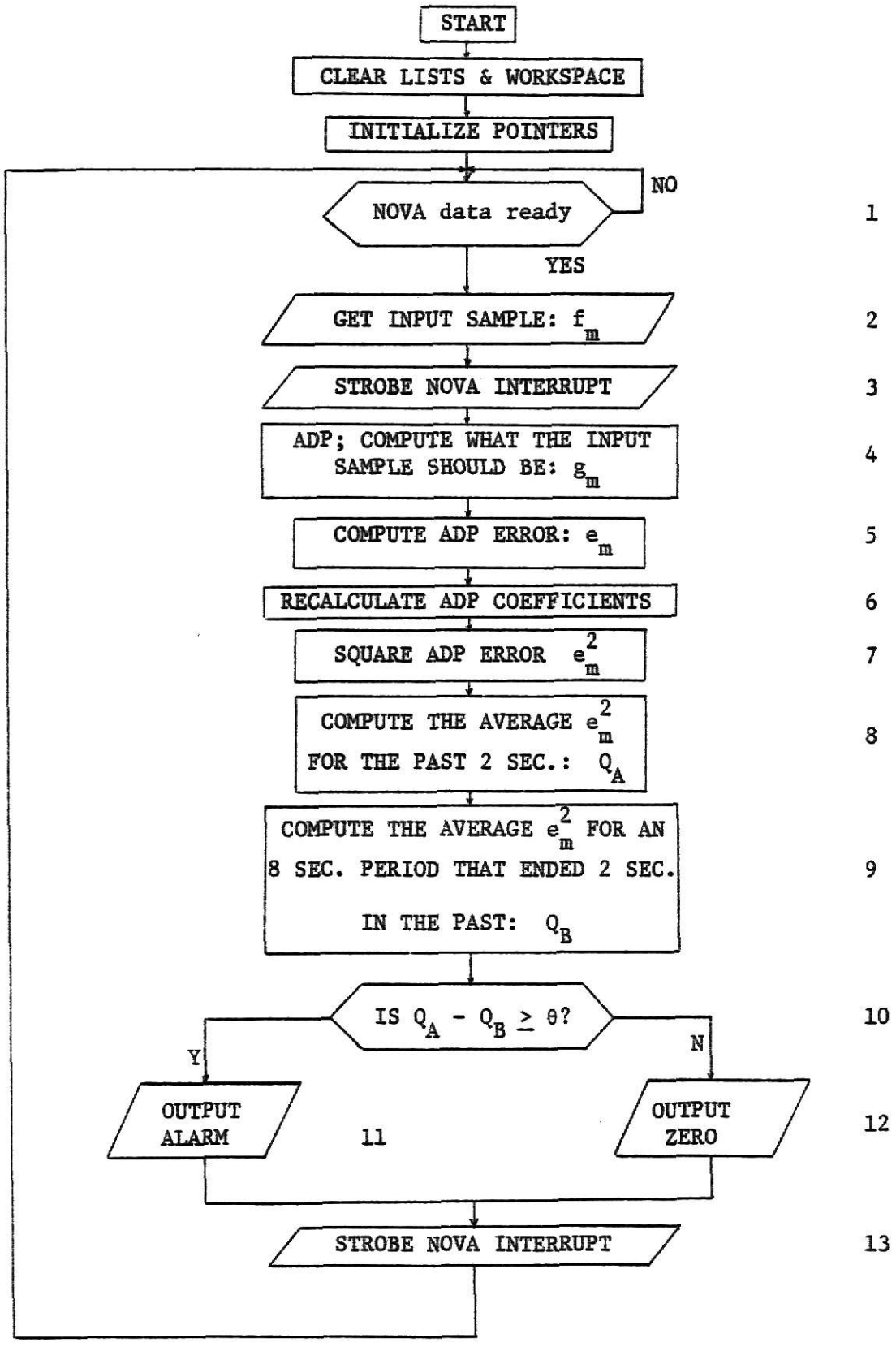

Fig. 6. Flowchart pertaining to implementation.

found to be less than one. Thus, the format of g_m is also set to (3/0/13). Since this format of g_m is not compatible with that of f_m, g_m is shifted left one bit to give it the format (2/0/13). The ADP error is then found in block 5 by negating g_m and adding f_m to $(-g_m)$. The error term has a (1/1/13) format.

In block 6, the coefficients b_k used by the ADP are recalculated. Each updated coefficient is the sum of two products. The first product is computed in two steps. The convergence parameter, v, is first multiplied by e_m. The format of v is set to (1/0/15), so that its hexadecimal representation, 1000_{16}, corresponds to the decimal fraction 0.125. The product $v * e_m$ has the format (2/1/13). The second step of this computation is to multiply $(v * e_m)$ by f_m. This product has the format (4/1/11), and is used in updating b_k as indicated in (2). The format of $\hat{u}$ is set to (4/0/12). Thus, the hexadecimal value of $\hat{u}$, $0E00_{16}$, is equal to the decimal fraction 0.875. Since the product $\hat{u} * b_k$ has the format (5/0/11), it can immediately be added to the $v * e_m * f_m$ product. The result has a (3/2/11) format, which does not agree with the format assumed for b_k in the ADP. Again, the careful assumption is made that b_k is less than unity, so that the format for these coefficients can be considered to be (5/0/11). Thus, shifting each updated b_k 4 bits to the left results in the desired format conversion.

Execution of the ATD algorithm routine begins at block 7. The squared value of e_m^2 is obtained by taking the absolute value of e_m and multiplying it by itself. The format of e_m^2 is now (2/2/12). In the interest of preserving precision later in the routine, e_m^2 is mentally shifted left to change its format to (1/2/12). In actuality, e_m^2 is

shifted 3 bits right, so that the result is effectively $e_m^2/16$ with the binary point three bits from the left.

Next, an estimate (say Q_A) of the variance term σ_j^2 in (9) is computed in block 8 of Fig. 6. To this end, the most recent value of $e_m^2/16$ is added to Q_A, and $e_{m-16}^2/16$ is then subtracted from the resulting Q_A. Thus, Q_A is continuously updated and can never overflow, since each of the 16 addends have been pre-divided by 16. A similar approach is used to compute Q_B in block 9, where Q_b is an estimate of σ_{j-D}^2 in (9).

Next, Q_B is subtracted from Q_A in block 10 of Fig. 6. No shifting is required in this connection, since Q_A and Q_B each have a (1/2/13) format. The threshold parameter θ is assumed to have a compatible format, (3/0/13), and hence can be directly subtracted from $(Q_A - Q_B)$. Thus, according to (9), if $(Q_A - Q_B - \theta)$ is negative, no intruder is present, and the alarm register is cleared. Conversely, $(Q_A - Q_B - \theta) \geq 0$ implies that an intruder is present. The alarm register is hence loaded with a large positive number, and is output in block 11 or 12 of Fig. 6. Next, the NOVA is signalled that an output is ready for storage. The program then returns to await a new input from the NOVA.

Experimental Results. The input data from the NOVA to the fixed point implementation of the overall algorithm, as described above, is shown in Fig. 7(a). The related sampling frequency is 8 samples per second (sps). This data was obtained on a test site via a special purpose digital data acquisition system. The response of the sensor (i.e., buried cable) was due to wind gusts. The symbol "↑" in Fig. 7(a) indicates an intruder crossing. Since the sampling is 8 sps, it follows

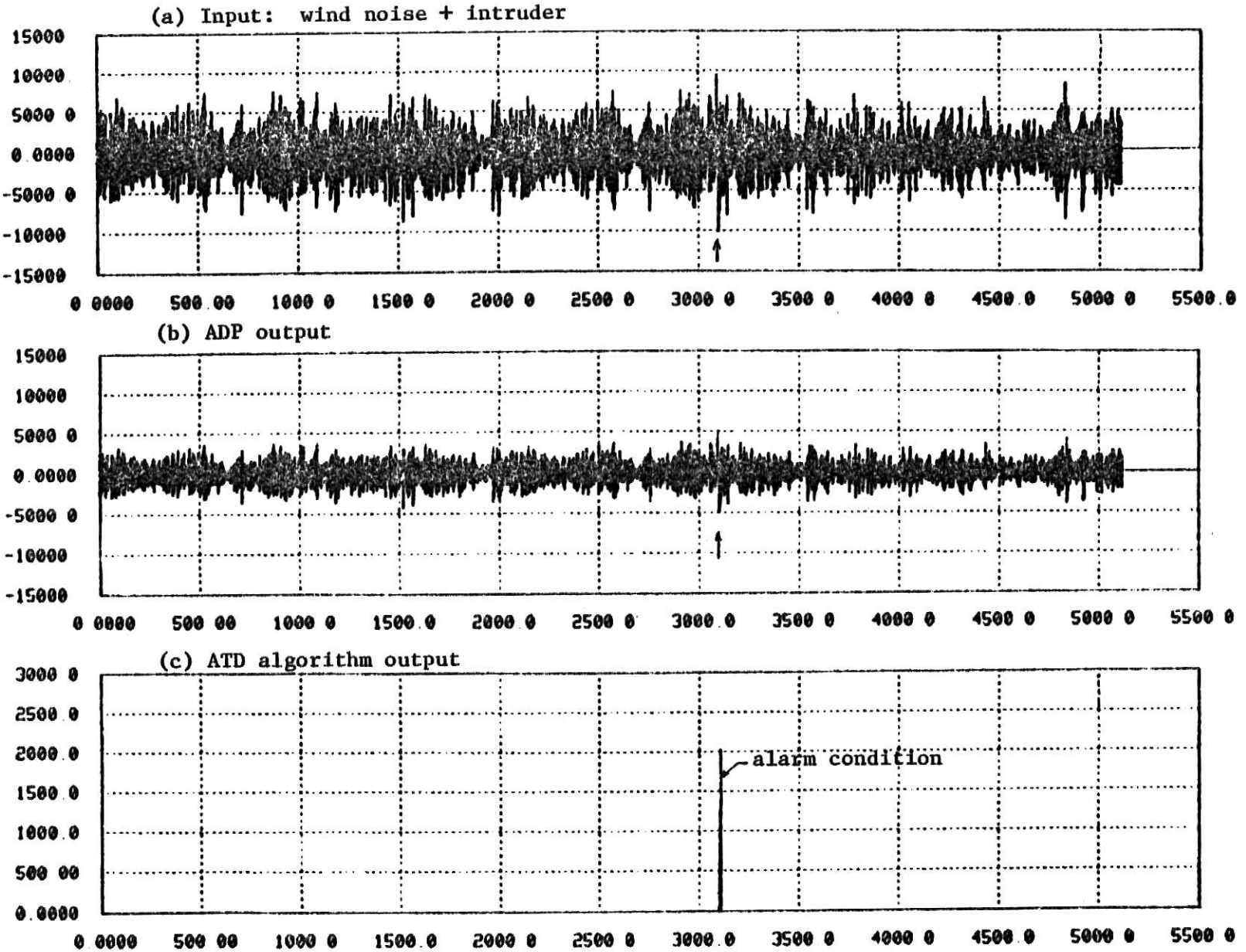

Fig. 7. Pertaining to experimental results: sampling frequency = 8 sps.

that this data file corresponds to approximately 10.6 minutes of real-time data. The object of the experiment is to detect the intruder crossing.

The corresponding output of a 16-weight ADP with $v = 0.125$ and $\hat{u} = 1 - u = 0.875$ is shown in Fig. 7(b). Again, Fig. 7(c) shows the output that results from the ATD algorithm with $L = 64$, $D = 16$, $M = 16$, $K = 1$, and $\theta = 0.016846$. From Fig. 7(c) it is apparent that the occurrence of the alarm condition concurs with the intruder crossing, which is the desired result.

VII. Concluding Remarks

The feasibility of implementing a specific intrusion-detection scheme on a commercially available 16-bit microcomputer system, using fixed point arithmetic has been demonstrated. Various details pertaining to this implementation were presented, and a useful notation for reducing overflow problems in a systematic manner was introduced. The author feels that this information can be used to good advantage in other adaptive digital signal processing applications in an on-line or real-time environment.

Future efforts related to this intrusion-detection approach will involve the possibility of implementing the ADP via lattice structure [3], in lieu of the transversal filter implementation considered here. This is because lattice structures have superior convergence properties, which is a feature that could enhance the performance of the ADP and the ATD algorithm discussed in this paper.

Acknowledgement

The author would like to thank the Sandia Laboratories, Albuquerque, N.M., for providing the data used in connection with the experimental results reported in this paper. The assistance and encouragement provided by Glenn Elliot, Jim Simpson and Dick Wayne, all of Sandia Laboratories, is gratefully acknowledged.

The author also wishes to express his appreciation to Dr. Nasir Ahmed and Dr. M. S. P. Lucas for serving as graduate committee members and for freely sharing their knowledge and experience. A special thanks is extended to my major advisor, Dr. Donald H. Lenhert, whose many teachings, both in and out of the classroom, have been an invaluable assistance when assistance was most desperately needed.

References

[1]. N. Ahmed, et al., "On an Intrusion-Detection Approach Via Adaptive Prediction," to appear in the IEEE Trans. on Aerospace and Electronic Systems.

[2]. B. Widrow, et al., "Adaptive Noise Cancelling: Principles and Applications," Proc. IEEE, Vol. 63, No. 12, pp. 1692-1716, Dec. 1975.

[3]. J. Makhoul, "A Class of All-Zero Lattice Digital Filters: Properties and Applications," IEEE Trans. Acoust. Speech, Signal Processing, Vol. ASSP-26, No. 4, pp. 304-314, Aug. 1978.

[4]. D. Godard, "Channel Equalization Using a Kalman Filter for Fast Data Transmission," IBM Journal of Research and Development, pp. 267-273, May 1974.

[5]. R. D. Gitlin and F. R. Magee, Jr., "Self-Orthogonalizing Adaptive Equalization Algorithms," IEEE Trans. on Communications, Vol. COM-25, No. 7, pp. 666-672, July 1977.

[6]. N. Ahmed, D. R. Hummels, and M. Uhl, "A Weighted Sequential Regression Algorithm," to appear in the IEEE Trans. on Communications.

[7]. N. Ahmed, G. R. Elliott, and S. D. Stearns, "Long-Term Instability Problems in Adaptive Noise Cancellers," Sandia Laboratories, Tech-Rept-No. SAND 78-1032, Aug. 1978; available from the National Technical Information Service, 5285 Port Royal Road, Springfield, VA 22161.

[8]. K. J. Hass and D. H. Lenhert, "Implementation of an Intrusion-Detection Algorithm on the 9900 Microcomputer," Progress Report, Dept. of Electrical Engineering, Contract #07-8254; November 1978.

[9]. N. Ahmed, G. R. Elliott, and S. D. Stearns, "Long-Term Stability Considerations of Adaptive Predictors," to appear in the 1978 Proc. of the Asilomar Conference on Circuits, Systems, and Computers.

[10]. M. Schwartz and L. Shaw, Signal Processing: Discrete Spectral Analysis, Detection, and Estimation, McGraw-Hill, Chap. 5, 1975.

[11]. A. Papoulis, Signal Analysis, McGraw-Hill, Inc., New York, 1977.

[12]. J. Simpson, "Fixed Point Implementation of Adaptive Digital Filters," to be published as a tech. rept. by Sandia Laboratories, Albuquerque, NM 87185.

ON A MICROCOMPUTER IMPLEMENTATION OF AN INTRUSION-DETECTION ALGORITHM

by

KENNETH JOSEPH HASS

B. S., Kansas State University, 1978

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1979

ABSTRACT

The main objective of this paper is to discuss various aspects of implementing a specific intrusion-detection scheme on a microcomputer system using fixed-point arithmetic. In particular, a TM 990/100M microcomputer system is considered.

The proposed scheme is suitable for detecting intruder stimuli which are in the form of transient signals. It consists of two stages: an adaptive digital predictor (ADP) and an adaptive threshold detection (ATD) algorithm. The ADP is used to remove correlated noise and hence reduce false (nuisance) alarms. It also yields an improvement in signal-to-noise ration when intruder stimuli are present, thereby assisting the ATD algorithm in their detection. The ATD algorithm is adaptive in the sense that it uses a pair of variances which are continually updated as the ADP output is processed. A function of these variances is compared with a preselected threshold for detection purposes.

Experimental results involving data acquired via field experiments are also included.