JOURNAL OF COMPUTATIONAL PHYSICS 2, 465–483 (1968)

Time-Domain Design of Finite-Memory Digital Filters[1]

T. A. BRUBAKER AND D. R. STEVENS

College of Engineering, University of Missouri, Columbia, Missouri 65201

ABSTRACT

Filtering is the process of transforming data from an experiment or process into a form which is more acceptable for analysis by a computer and/or a person. In the past, most filtering has been accomplished by use of analog devices; however, the increasing speed and decreasing size and cost of digital components now permits the economical use of digital filters to perform many desired filtering operations.

In this paper, three methods for time domain design of finite-memory digital filters are presented. These filters can perform the operations of smoothing, prediction, and differentiation either separately or simultaneously and are easily implemented by programming on small digital computers or by a hard-wired design using digital logic. In addition the use of infinite-memory digital filters and problems encountered in their time-domain design are mentioned.

INTRODUCTION

In experimental research, it is often necessary to transform experimental data into a form which is more useful to the researcher for evaluation either by himself or with help from modern computers. In the past, this has usually been accomplished by use of analog filters whose design is based on well-known theory. Within the past several years, however, the increasing speed and decreasing size and cost of digital computer components have permitted the economical use of digital filters to provide the desired transformations in real time.

Real-time digital filters have several advantages over continuous analog filters. First, a greater degree of accuracy can be obtained in the filter realization because the accuracy is dependent on the word length and sampling rate of the filter. Since these variables are controlled by the designer, the transformations can be as accurate as desired. This is opposed to analog filters whose accuracy is highly dependent on the components used in their design. Second, a greater variety of

[1] This work was supported by Lawrence Radiation Laboratory under a contract for the study of Open and Closed Loop Techniques for Computer Controlled Instrumentation.

digital filters can be built, and with computers used for filtering, filter designs can be changed by programming which allows one computer to be used for a large number of filter designs. Finally, no special components are needed to realize filters with time varying coefficients. As a result, digital filters are finding their way into an increasing number of applications, particularly when a digital computer is already available for use in an experiment or process for data logging and/or control.

General Digital Filter Theory

Linear digital filter theory is based on the well-known mathematics of linear difference equations. The general form of a linear digital filter is expressible as

$$y(nT) = \sum_{K=0}^{N-1} a_k x[(n-k)\,T] - \sum_{j=1}^{M-1} b_j y[(n-j)\,T], \tag{1}$$

where the $\langle a_k, b_j \rangle$ are either constants of functions of the independent variable nT. For this paper, $\langle a_k, b_j \rangle$ are assumed to be constants. In (1), and elsewhere in this paper, $y[nT]$ denotes the present output and $y[(n-j)\,T]$, $j = 1, 2,\dots M-1$ denotes past filter outputs. The terms $X[(n-K)\,T]$, $K = 0, 1, 2,\dots N-1$ denote present and past inputs to the filter. The symbol nT is used to indicate a discrete computation or sampling interval where T is the length of the interval and n is an integer. For simplicity and to comply with notation used in most difference-equation theory, (1) can be rewritten as

$$y_n = \sum_{K=0}^{N-1} a_k x_{n-k} - \sum_{j=1}^{M-1} b_j y_{n-j}. \tag{2}$$

From a practical point of view, the discrete inputs are assumed to come from an analog-to-digital converter. If quantization effects are neglected, the A/D converter output for a constant sampling rate is

$$\begin{aligned} x^*(t) &= x(nT),\ nT \leqslant t < (n+1)\,T, \\ n &= 0, 1, 2, 3,\dots, \end{aligned} \tag{3}$$

where $x(t)$ is the continuous input signal, usually a voltage, and $x^*(nT)$ is the value of $x(t)$ at time $t = nT$. In (3) it is assumed that $x(t)$ starts at time $t = 0$.

The output of the digital filter $y_n = y(nT)$ can appear in two ways. First, y_n can appear simply as a number on a digital printout. This number must then be related to the range of the variable being studied. Second, y_n can appear as the output of a digital-to-analog converter. This variable, which is discrete, can then be used directly for display or control.

A block diagram of a computer used for filtering and a pictorial view of the discrete input and output is shown in Fig. 1. Since past inputs and outputs are also used in general filter calculations, it is necessary that a digital memory be available for storage.

In time-domain design of digital filters, (2) provides a direct starting point. If, however, the frequency response of the filter is desirable, or if frequency-response characteristics are used for design, it is necessary to find the Laplace and Z trans-

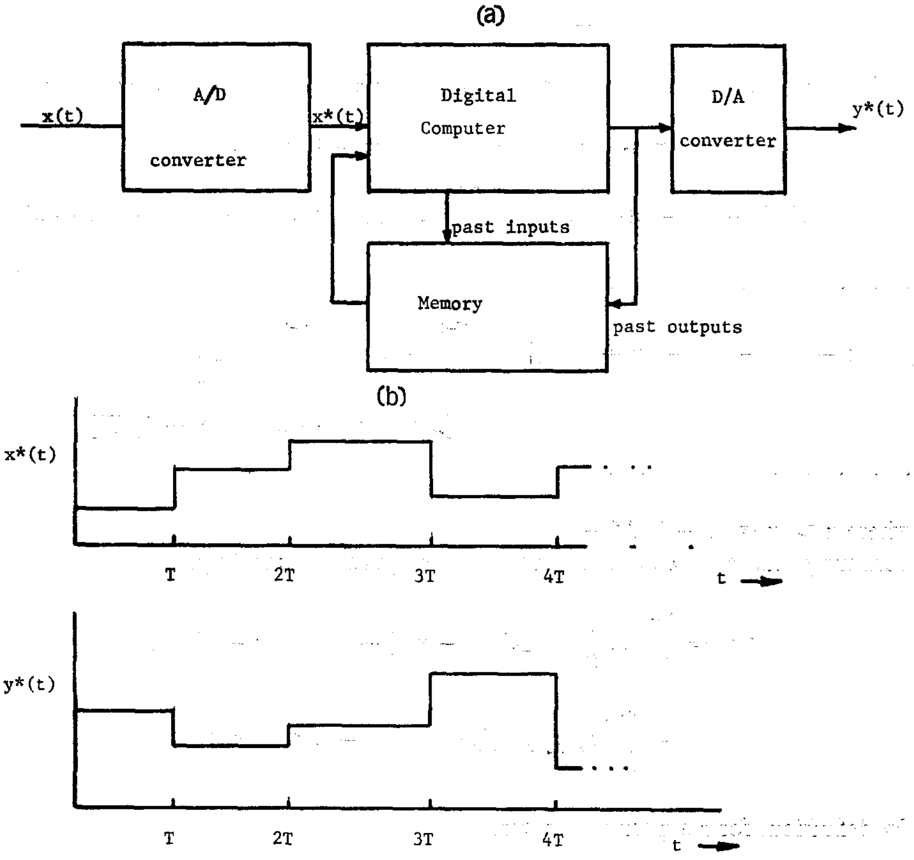

FIG. 1. (a) Block diagram of a computer used for digital filtering; (b) a typical Input–Output sequence for a digital filter.

form of (2). This is accomplished as follows. The actual digital computer input and output time functions can be written as

$$x^*(t) = \sum_{n=0}^{\infty} x(nT)[\mu[t - nT] - \mu[t - (n+1)T]],$$
$$y^*(t) = \sum_{n=0}^{\infty} y(nT)[\mu[t - nT] - \mu[t - (n+1)T]], \tag{4}$$

where

$$\begin{aligned} \mu(t - nT) &= 1, \qquad t \geqslant nT, \qquad n = 0, 1, 2, \ldots, \\ &= 0, \qquad t < nT. \end{aligned}$$

Referring to (2) for the input $x(t)$ starting at time $t = 0$, it is easily seen that

$$y_0[\mu(t) - \mu(t - T)] = a_0 x_0[\mu(t) - \mu(t - T)],$$
$$y_1[\mu(t - T) - \mu(t - 2T)] = \left[\sum_{k=0}^{1} a_k x_{1-k} - b_1 y_0\right][\mu(t - T) - \mu(t - 2T)],$$
$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots, \tag{5}$$
$$y_n[\mu(t - nT) - \mu[t - (n+1)T]] = \left[\sum_{k=0}^{N-1} a_k x_{n-k} - \sum_{j=1}^{M-1} b_j y_{n-j}\right] \times [\mu(t - nT) - \mu[t - (n+1)T]]$$
$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots,$$

where $n > \max N - 1, M - 1$.
Summing this set of equations and letting n approach infinity gives

$$\begin{aligned} y^*(t) = &\sum_{k=0}^{N-1} a_k \sum_{n=k}^{\infty} X_{n-k}[\mu[t - nT] - \mu[t - (n+1)T]] \\ &- \sum_{j=1}^{M-1} b_j \sum_{n=j}^{\infty} y_{n-j}[\mu[t - nT] - \mu[t - (n+1)T]] \end{aligned} \tag{6}$$

By definition, for r a positive integer,

$$\begin{aligned} x^*(t - rT) &= 0, \\ y^*(t - rT) &= 0, \end{aligned} \qquad t < rT \tag{7}$$

and

$$\left.\begin{aligned} x_{n-k} &= 0, \\ y_{n-k} &= 0, \end{aligned}\right\} \quad n < k, \tag{8}$$

so that

$$\begin{aligned} y^*(t - rT) &= \sum_{n=r}^{\infty} y_{n-r}[\mu[t - nT] - \mu[t - (n+1)\,T]], \\ x^*(t - rT) &= \sum_{n=r}^{\infty} x_{n-r}[\mu[t - nT] - \mu[t - (n+1)\,T]]. \end{aligned} \tag{9}$$

Using results shown in (9), (6) can be rewritten as

$$y^*(t) = \sum_{k=0}^{N-1} a_k x^*(t - kT) - \sum_{j=1}^{M-1} b_j y^*(t - jT). \tag{10}$$

Using the Laplace transform relationships

$$\begin{aligned} \mathscr{L}\{g(t)\} &= G(s), \\ \mathscr{L}\{g(t - kT)\} &= G(s)\, e^{-ksT} \end{aligned} \tag{11}$$

and taking the Laplace transform of (10) yields the transfer function of a digital filter

$$\frac{Y^*(s)}{X^*(s)} = \frac{\sum_{k=0}^{N-1} a_k e^{-ksT}}{1 + \sum_{j=1}^{M-1} b_j e^{-jsT}}. \tag{12}$$

Letting $z = e^{sT}$ now gives the z transform of the digital filter transfer function as

$$\frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{N-1} a_k z^{-k}}{1 + \sum_{j=1}^{M-1} b_j z^{-j}} \tag{13}$$

Equation (12) can be used to find the frequency and phase response of a digital filter by letting $s = j\omega$ and plotting the absolute value and angle of the resulting transfer function vs ω. It should be carefully noted that the presence of the exponential functions in (12) can give a magnitude response which is periodic in ω with period $2\pi/T$ where T is the sampling interval. This gives rise to the concept of aliasing errors which means that if the sampling rate is not high enough, high-frequency components of a sampled signal appear to be the same at the analog-to-digital converter output as low-frequency signals. This can be avoided by making the sampling frequency at least two times as high as the highest-frequency component of the signal which is to be sampled.

Equation (12) is used for certain aspects of design and is used when Z-transform methods are used to analyze complete digital or sample-data control systems. In such systems, the digital computer is only one of a number of components in the overall system.

The following discussion deals with time domain design of digital filters with polynomial inputs. From a conceptual point of view, this type of design is closely related to classical methods of data smoothing which use polynomials to fit experimental data. The advantages are that digital computers are well-suited for operating on polynomial input functions. The main disadvantage of time-domain design occurs when frequency characteristics of input signals are of prime interest. In this case, it is often best to consider frequency-domain analysis which is well described in a recent paper by Radar and Gold [1].

Time-Domain Design of Finite-Memory Digital Filters

A linear finite-memory digital filter can be represented by the equation

$$y_n = \sum_{k=0}^{N-1} a_k x_{n-k}, \tag{14}$$

where y_n represents the output at time $t = nT$, and x_{n-k} are inputs at times $(n-k)T$, $k = 0, 1, 2, \ldots N-1$. The a_k will be assumed to be constants.

Digital filters represented by (14) can perform the operations of

1. Smoothing
2. Prediction
3. Differentiation

either separately or simultaneously. These filters operate on a predetermined number of the input samples and for simplicity, the span of the input samples are usually considered to be a data window. A typical data window is shown in Fig. 2. Note that the window moves down the time axis at the sampling rate and that (14) can, for this reason, be considered a form of convolution. Also note that, for an input signal starting at $t = 0$, or for abrupt changes in the signal for $t > 0$, it takes $(N - 1)$ sampling times for all of the data-window points to be aware of the change. Thus, the desired filter operation has a transient response which lasts $(N - 1)$ sampling periods. This type of operation has no analog counterpart since

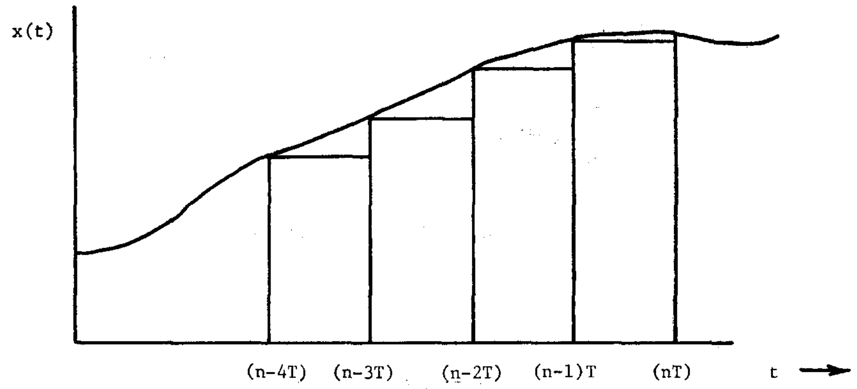

FIG. 2. Typical data window for a finite-memory digital filter.

analog filters have a transient response due to discontinuities, which possess infinite settling time.

The input to the finite-memory digital filters will be assumed to be a polynomial signal plus noise. As in classical theory, for nonpolynomial signals it will be assumed that, over a finite-size data window, the input signal can be reasonably approximated by a polynomial. The digital filter will then be designed to provide the desired transformation while, at the same time, reducing the effect of input noise as much as possible. For simplicity, the noise will be assumed to have zero mean value, variance σ^2, and zero correlation between samples.

Finite-Memory Digital Filters for Smoothing and Prediction

In general, the operation of smoothing is used to improve the signal-to-noise ratio of a signal corrupted by noise. The operation of prediction is used primarily for control purposes in digital feedback systems. In both operations, the desired filter output is some function of the polynomial input. In practice this is impossible to achieve; one can only reduce the noise, not eliminate it. For an input $x(t)$, consisting of a polynomial $p(t)$ of order q, the desired form of a smoothing and/or prediction filter is $y^*(t) = p^*(t - \alpha T)$. From a computer point of view $y_n = p_{n-\alpha}$ so that (14) can be written as

$$p_{n-\alpha} = \sum_{k=0}^{N-1} a_k p_{n-k} \,. \tag{15}$$

The reader should carefully observe that, for α positive, the output lags the input, while for α negative, the output leads or predicts the input.

Expanding p_{n-k} and $p_{n-\alpha}$ in Taylor series about the point nT and collecting like terms results in a set of necessary conditions on the constants a_k.

$$\sum_{k=0}^{N-1} a_k = 1,$$
$$\sum_{k=0}^{N-1} k^r a_k = \alpha^r, \qquad r = 1, 2, 3, 4, \ldots, q. \tag{16}$$

If this set of simultaneous equations is to have a solution, the relationship between N and q must be

$$N - 1 \leqslant q.$$

However, if we want to choose some of the a_k in accordance with other design criterion, we will generally assume that

$$q < N - 1.$$

When $x(t)$ is a polynomial signal $p(t)$ plus noise,

$$x(t) = p(t) + n(t), \tag{17}$$

the expected value of the filter output is

$$E\{y_n\} = \sum_{k=0}^{N-1} a_k E\{p_{n-k} + n_{n-k}\} = \sum_{k=0}^{N-1} a_k p_{n-k}\,. \tag{18}$$

This means that when the polynomial signal $p(t)$ is corrupted by additive noise $n(t)$, the expected value of the filter output is due to signal only.

The variance of the filter output is

$$\begin{aligned}\operatorname{Var}\{y_n\} &= E\{y_n - E\{y_n\}\}^2 \\ &= E\left\{\sum_{k=0}^{N-1} a_k[p_{n-k} - n_{n-k}] - \sum_{k=0}^{N-1} a_k p_{n-k}\right\}^2 \\ &= \sigma^2 \sum_{k=0}^{N-1} a_k{}^2\end{aligned} \tag{19}$$

and it is this function that is to be minimized if the output noise is to be reduced as much as possible.

The design problem is now easily stated. Given a filter input consisting of a

qth-order polynomial $p(t)$ plus noise $n(t)$, if the desired output at the nth sampling period is $y_n = p_{n-\alpha}$, then the function described by (19) must be minimized with respect to the a_k subject to the constraints of (16). The resulting a_k will give the desired sample weights for a finite-memory digital filter.

This problem can be handled by use of Lagrange multipliers, least-squares theory, and least-squares theory using orthogonal polynomials.

Design Using Lagrange Multipliers

To minimize the function

$$\frac{\operatorname{Var}\{y_n\}}{\sigma^2} = \sum_{k=0}^{N-1} a_k{}^2$$

subject to the constraints of (16), we minimize the new function

$$f(\lambda_r, a_k) = \sum_{k=0}^{N-1} a_k{}^2 + \sum_{k=0}^{N-1} \sum_{r=0}^{q} 2\lambda r[k^r a_k] - \sum_{r=0}^{q} 2\lambda_r \alpha^r \tag{20}$$

with respect to the a_k. In (20), the λj are called Lagrange multipliers. Taking the derivatives of $f(\lambda_r, a_k)$ with respect to the a_k and setting the results equal to zero results in a set of equations,

$$\begin{gathered} a_k + \lambda_0 + k\lambda_1 + k^2\lambda_2 + \cdots k^{N-1}\lambda q = 0, \\ k = 0, 1, 2, \ldots N-1. \end{gathered} \tag{21}$$

By combining (21) and (16) there results a set of $N + q + 1$ simultaneous equations in $N + q + 1$ unknowns. The solution of these equations give the λi, $i = 0, 1, \ldots q$, which in turn can be used to find the a_k, $k = 0, 1 \ldots N - 1$.

Example. For $q = 0$, $N - 1 = 3$, and $\alpha = 0$, the constraints are

$$a_0 + a_1 + a_2 + a_3 = 1 \tag{22}$$

and the set of simultaneous equations resulting from minimization are

$$a_k + \lambda_0 = 0, \qquad k = 0, 1, 2, 3. \tag{23}$$

Solving (22) and (23) yields $\lambda_0 = \frac{1}{4}$, $a_0 = a_1 = a_2 = a_3 = \frac{1}{4}$
The resulting digital filter is

$$y_n = \frac{x_n + x_{n-1} + x_{n-2} + x_{n-3}}{4} \tag{24}$$

and the variance of the output is easily seen to be

$$\mathrm{Var}\{y_n\} = \sigma^2/4. \tag{25}$$

It is worth noting that this example is similar to use of ensemble averaging of repeated experiments for signal-to-noise enhancement [2].

Example. for $q = 1$, $N - 1 = 3$, and $\alpha = 0$, the constraints are

$$\begin{aligned} a_0 + a_1 + a_2 + a_3 &= 1, \\ a_1 + 2a_2 + 3a_3 &= 0 \end{aligned} \tag{26}$$

and the equations resulting from minimization of (20) are

$$\begin{aligned} a_0 + \lambda_0 &= 0, \\ a_1 + \lambda_0 + \lambda_1 &= 0, \\ a_2 + \lambda_0 + 2\lambda_1 &= 0, \\ a_3 + \lambda_0 + 3\lambda_1 &= 0. \end{aligned} \tag{27}$$

Solving (26) and (27) results in a digital filter

$$y_n = \frac{7x_n + 4x_{n-1} + x_{n-2} - 2x_{n-3}}{10} \tag{28}$$

whose output variance is

$$\mathrm{Var}\{y_n\} = 7\sigma^2/10 \tag{29}$$

Design Using Least-Squares Theory

Use of least-squares theory results in the same infinite-memory digital filters for a given order input, as does the Lagrange multiplier method. To see this, first assume that the data window starts at time $t = 0$. (Since the value of the filter weights is invariant under time transformation this will cause no problem.) Using least-squares theory, the polynomial form of the filter output is assumed and the function

$$R = \sum_{k=0}^{N-1} [y_k - x_k]^2 = \sum_{k=0}^{N-1} [b_{q0} + b_{q1}(KT) + b_{q2}(kT)^2 + b_{qq}(kT)^q - x_k]^2 \tag{30}$$

is minimized with respect to the y polynomial coefficients. The solution of this set of equations results in coefficients which are functions of the input samples x_k. Substitution of the coefficients back into the polynomial then results in an equation which yields the filter weights for a specified data window.

Since the general theory of least-square-curve fitting is well described in the literature [3], only examples will be given to illustrate how to use the theory in filter design.

Example. For a four-point data window and

$$y(t) = b_{10} + b_{11}t,\ y_n = b_{10} + b_{11}(nT), \quad (30)$$

becomes

$$R = \sum_{k=0}^{3} [b_{10} - b_{11}kT - x_k]^2. \tag{31}$$

Minimizing R with respect to b_{10} and b_{11} results in two equations,

$$\begin{aligned} 4b_{10} + 6Tb_{11} &= x_0 + x_1 + x_2 + x_3\,, \\ 6Tb_{10} + 14T^2b_{11} &= Tx_1 + 2Tx_2 + 3Tx_3 \end{aligned} \tag{32}$$

whose solution gives

$$\begin{aligned} b_{10} &= \frac{7x_0 + 4x_1 + x_2 - 2x_3}{10}\,, \\ b_{11} &= \frac{-3x_0 - x_1 + x_2 + 3x_3}{10T}\,. \end{aligned} \tag{33}$$

Substituting (33) into the expression for $y_n = b_{10} + b_{11}nT$ gives

$$y_3 = \frac{7x_3 + 4x_2 + x_1 - 2x_0}{10}\,. \tag{34}$$

Thus, for a general four-point data window and a first-order-polynomial fit,

$$y_n = \frac{7x_n + 4x_{n-1} + x_{n-2} - 2x_{n-3}}{10}\,. \tag{35}$$

This is exactly the same result that was obtained using Lagrange multipliers and a first-order-polynomial input.

It should be carefully observed that, in both design methods, if the input data is not well-approximated by the chosen polynomial over a data window, large errors can result which are not due to noise. The errors tend to distort input data vastly and should be avoided as much as possible.

Design Using Orthogonal Polynomials

The previous design methods unfortunately involve the solution of simultaneous equations which generally imply matrix inversion. To eliminate this, orthogonal polynomials can be used in the least-squares design. This method has the advantage of being easily and quickly done on a computer which is useful if filter designs are to be changed on-line so as to adapt to various input conditions.

Referring to (30), if the input polynomial $y(t)$ has the form

$$y(t) = b_{q0}\omega_0(t) + b_{q1}\omega_1(t) + \cdots b_{qq}\omega_q(t), \tag{36}$$

where the $\omega i(nT)$ are orthogonal when summed over the data window, then minimization of (30) with respect to the b_{qj}, $j = 0, 1, 2, \ldots q$, results in the coefficient equations

$$b_{qj} = \frac{\sum_{k=0}^{N-1} \omega j(kT)\, x(kT)}{\sum_{k=0}^{N-1} [\omega j(kT)]^2}, \qquad j = 0, 1, 2, \ldots, q \tag{37}$$

For the previous example, if

$$y = b_{10}\omega_0(t) + b_{11}\omega_1(t), \tag{38}$$

where

$$\omega_0(t) = 1, \qquad \omega_1(t) = t - \tfrac{6}{4}\, T, \tag{39}$$

then it is easily seen that

$$\sum_{k=0}^{3} \omega_0(kT)\, \omega_1(kT) = 0. \tag{40}$$

Substituting into (37) gives

$$\begin{aligned} b_{10} &= \frac{x_0 + x_1 + x_2 + x_3}{4}, \\ b_{11} &= \frac{-3x_0 - x_1 + x_2 + 3x_3}{10T} \end{aligned} \tag{41}$$

Substituting (41) into (38) gives a design which is the same as shown in (34).

The only problem in working with orthogonal polynomials is the determination of polynomials which are orthogonal over a given data window. Since most books discuss orthogonality in terms of integration over an interval from zero to one, it is necessary to set up a recursion formula which gives polynomials which have the desired properties. For readers interested in doing this, a report on this subject, which contains computer programs, has been written by Peterson [4].

Finite-Memory Filters for Differentiation, Prediction, and Smoothing

If the operations of differentiation, prediction, and smoothing are required, for an input polynomial approximation $p(t)$, the desired output is

$$y_n = \dot{p}_{n-\alpha}\,. \tag{42}$$

For Lagrange-multiplier design, this leads to a set of constraints

$$\sum_{k=0}^{N-1} a_k = 0,$$

$$\sum_{k=0}^{N-1} k^r a_k = \frac{-r\alpha^{r-1}}{T}, \qquad r = 1, 2, 3, \ldots, q \tag{43}$$

The remainder of the design works exactly the same as before.

In the case of least-squares smoothing, one simply takes the derivative of the polynomial which is fit to the data points and substitutes in the values for the polynomial coefficients found by considering smoothing and prediction.

Example. For $\alpha = 0$, $N = 4$, and $y(t) = b_{10} + b_{11}t, \dot{y}(t) = b_{11}$. Thus the filter design for differentiation and smoothing, but not prediction is

$$y_n = b_{11} = \frac{3x_n + x_{n-1} - x_{n-2} - 3x_{n-3}}{10T} \tag{44}$$

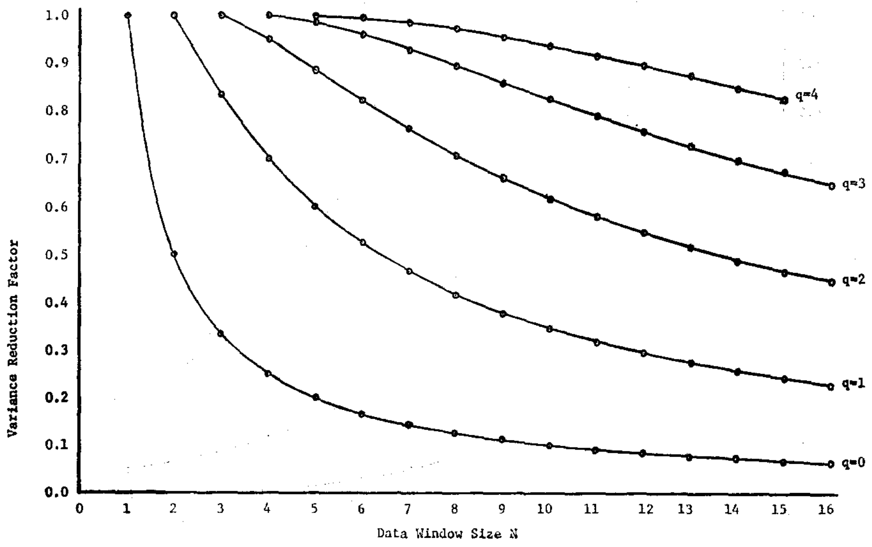

FIG. 3. Variance reduction factor plotted vs. data window size for a smoothing filter with polynomial inputs of order 0, 1, 2, 3, 4; $\alpha = 0$.

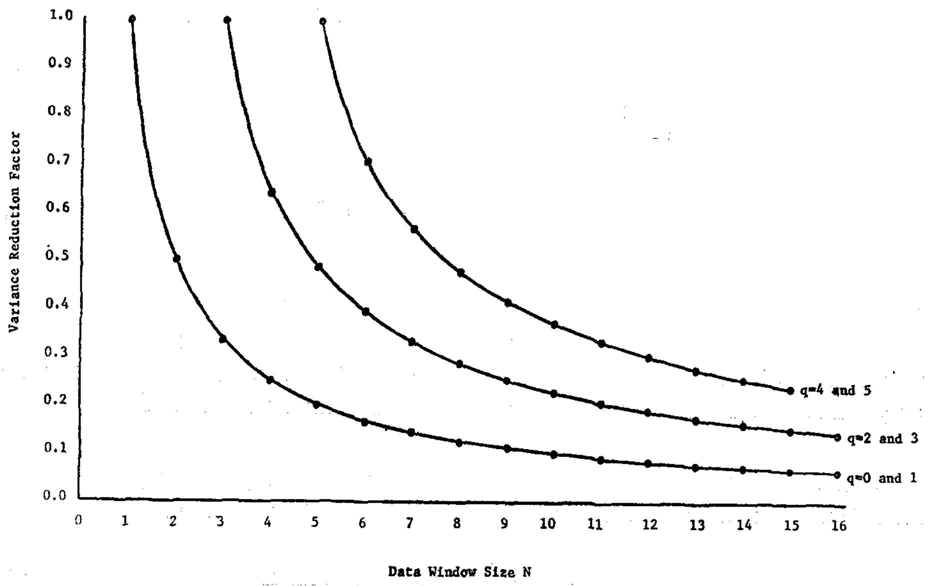

FIG. 4. Variance reduction factor plotted vs. data window size for smoothing filters with $\alpha = (N - 1)/2$ = center of the data window.

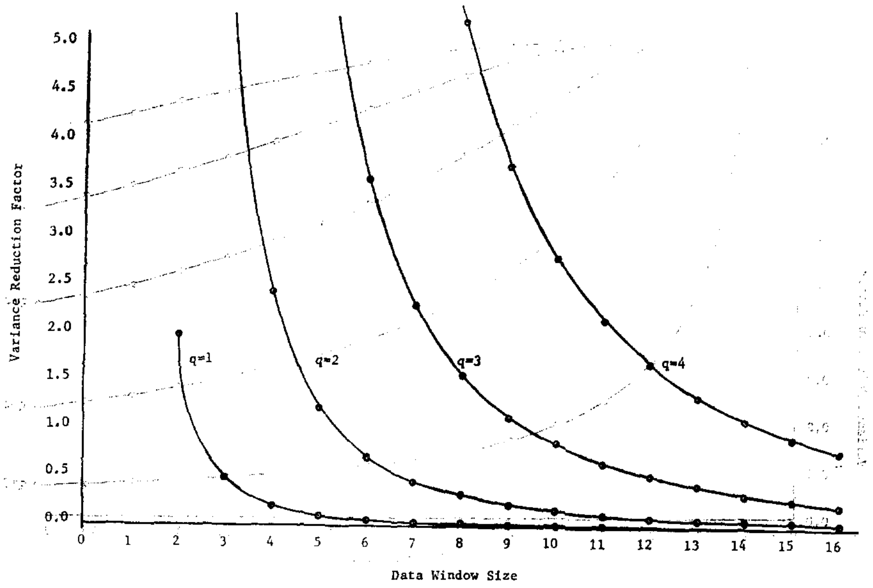

FIG. 5. Variance reduction factor plotted vs. data window size for differentiating plus smoothing; $\alpha = 0$. (Note that it is easy to introduce more noise into a differentiating filter.)

Design Curves for Finite-Memory Digital Filters

Figures 3–6 are plots of the variance of the filter output over the variance of the noise, plotted vs the data-window size for various types of filters with polynomial inputs. The interesting thing about these curves is that, for higher-order smoothing, a large data window must be used if significant signal-to-noise enhancement is to take place.

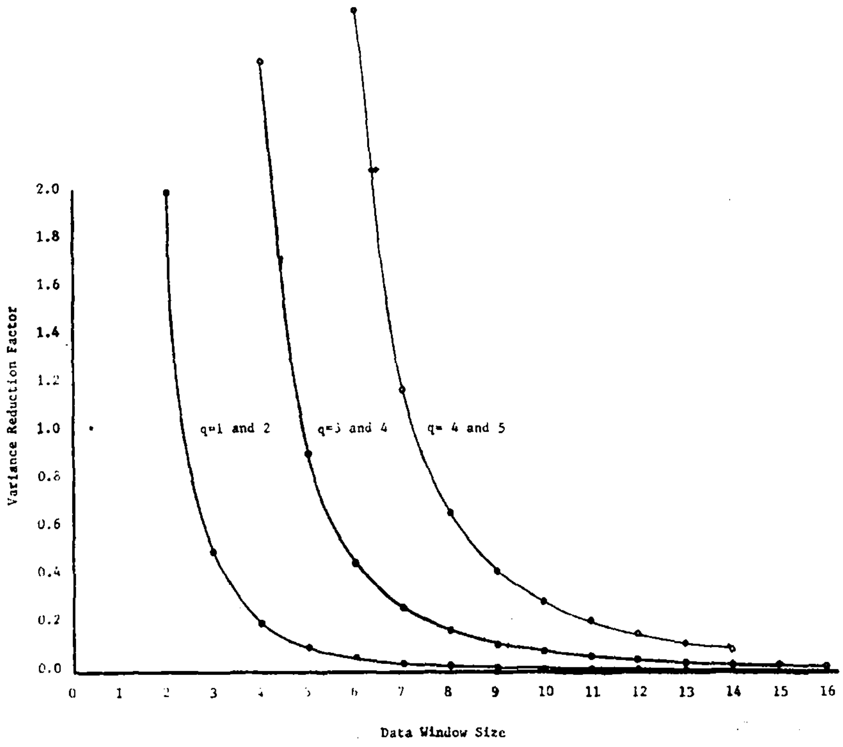

FIG. 6. Variance reduction factor plotted vs. data window size for differentiating smoothing and $\alpha = (N - 1)/2$.

Figures 7–11 are plots of the variance reduction, $\mathrm{Var}(y)/\sigma^2$, plotted vs β, where β is defined as

$$\beta = -\alpha + (N - 1)/2$$

These curves show that for a given polynomial input and given data window, there exist positive values of α which yield a maximum of signal-to-noise enhancement. It should be carefully noted that higher-degree polynomial inputs cause more

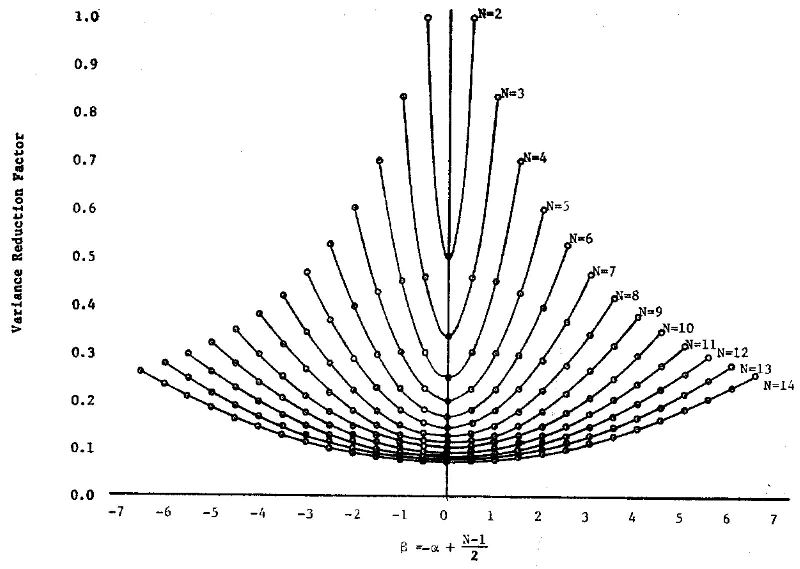

FIG. 7. Variance reduction factor plotted vs. $\beta = -\alpha + (N-1)/2$ for a smoothing filter with first-degree polynomial input. Curves are only shown for positive α.

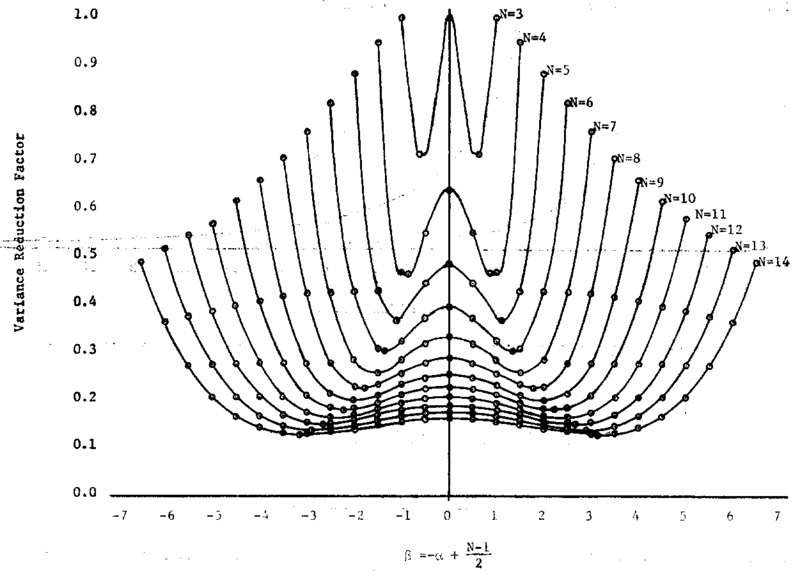

FIG. 8. Variance reduction factor plotted vs. $\beta = -\alpha + (N-1)/2$ for a smoothing filter with second-order polynomial input. Curves are only shown for positive α.

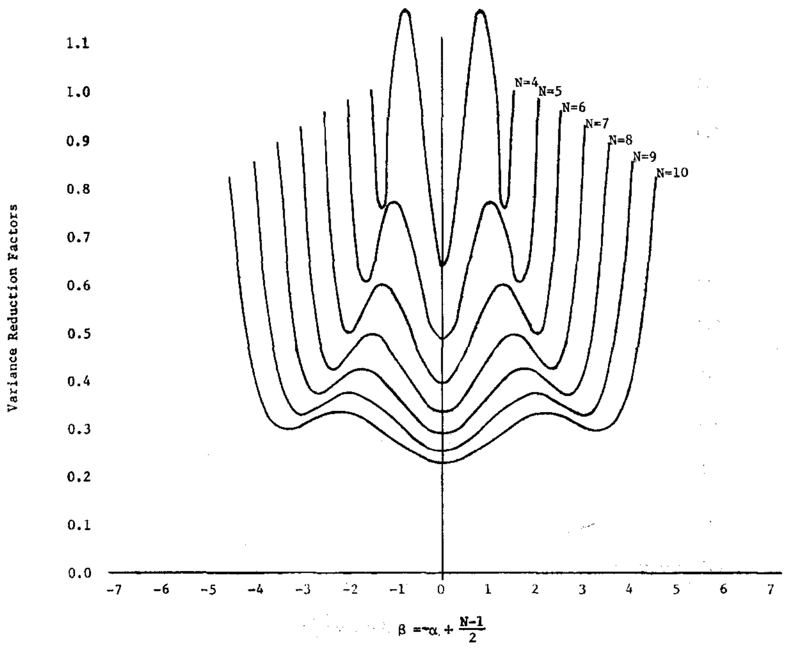

FIG. 9. Variance reduction factor plotted vs. $\beta = -\alpha + (N - 1)/2$ for a smoothing filter with a third-order polynomial input. Curves are only shown for positive α.

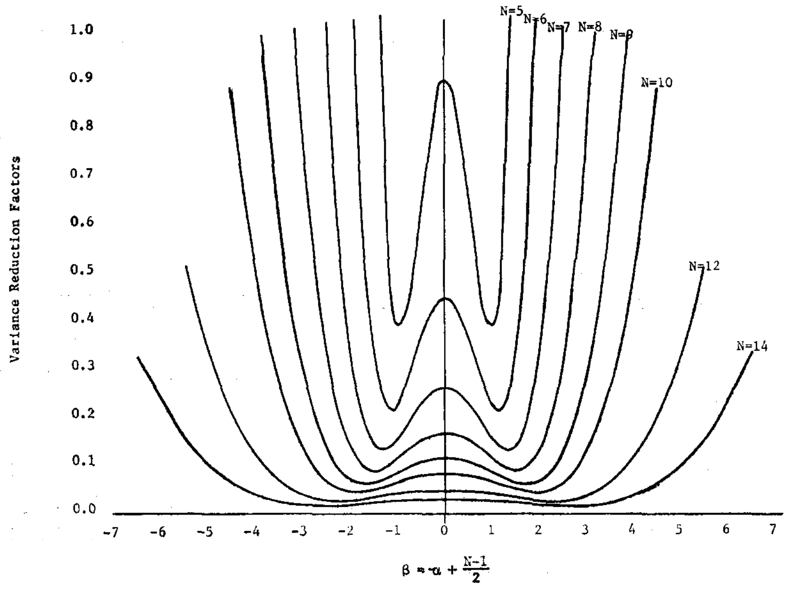

FIG. 10. Variance reduction factor plotted vs. $\beta = -\alpha + (N - 1)/2$ for differentiating and smoothing filter with third-order polynomial input. Curves are only shown for positive α.

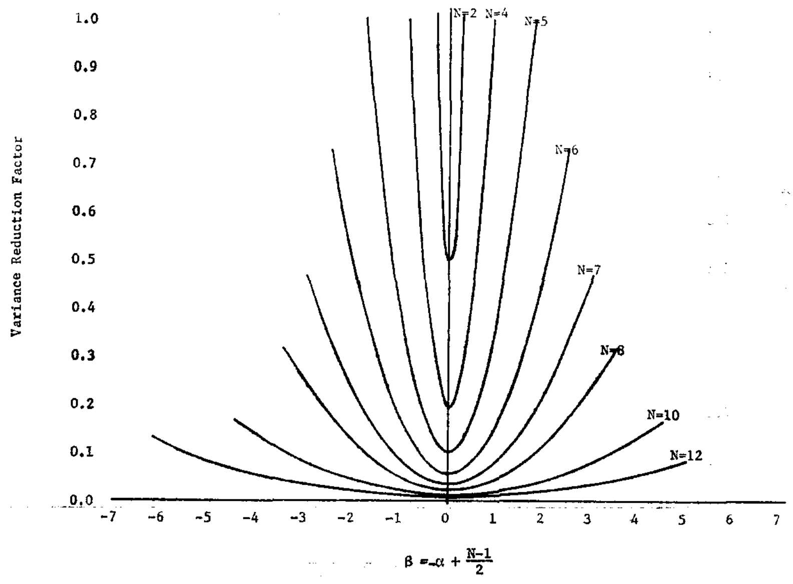

FIG. 11. Variance reduction factor plotted vs. $\beta = -\alpha + (N-1)/2$ for a differentiating and smoothing filter with second-order polynomial input. Curves are only given for positive α.

relative minima to occur and to get maximum smoothing, one should pick a value of β which gives a minimum and then solve for α. Readers should also be aware that positive values of α cause the filter to produce transport delay into the data, and if the filter is used for control, this delay will generally cause the control system to be less stable. Negative values of α, on the other hand, allow prediction but tend to introduce noise into the system as the magnitude of α increases.

TIME-DOMAIN SYNTHESIS OF INFINITE-MEMORY DIGITAL FILTERS

Infinite-memory digital filters can perform all of the operations that can be done with finite-memory digital filters plus the operation of integration. Except for the case of integration, however, a general method of time-domain design is not known. This is primarily due to the presence of the b_j terms in (2) which do not allow a straightforward minimization procedure. One possible solution to the problem appears to be a design using the ideas of state space, however, at present nothing definite can be said concerning this.

Even so, infinite-memory digital filters can be used; and the reader is referred to References [5]–[7] for information on presently-known design methods. These filters do not possess a finite settling time to changes in input signal, but they do yield a greater economy of memory space in a computer than do finite-memory filters for the same degree of smoothing.

References

1. C. M. Radar, and B. Gold, *Proc. IEEE*, February, 149–170 (1967).
2. M. P. Klein and G. W. Barton, Jr., *Rev. Sci. Instr.* **34**, 754–759 (1963).
3. R. W. Hamming, "Numerical Methods for Scientists and Engineers." McGraw-Hill, New York, 1962.
4. J. Peterson, Use of Orthogonal Polynomials in Digital Filter Design, Information Systems Simulation Laboratory Report, No. 2, August, 1967.
5. A. J. Monroe, "Digital Processes for Sampled Data Systems." Wiley, New York, 1962.
6. H. Holtz and C. T. Leondes, *J. ACM* **13**, 262–280 (1966).
7. F. F. Kuo and J. F. Kaiser (Eds.), "System Analysis by Digital Computer." Wiley, New York, 1966.