AFIT/GA/ENY/91D-15

Discrete Simulation of Fractional Order Systems

THESIS

Jeffrey A. Blank

AFIT/GA/ENY/91D-15

AFIT/GA/ENY/91D-15

Discrete Simulation of Fractional Order Systems

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Astronautical Engineering

Jeffrey A. Blank, B.S. in Astronautical Engineering

Captain, USAF

December 1991

Approved for public release: distribution unlimited.

Acknowledgments

This thesis reflects the influence of many people. First and foremost, I want to thank my advisor, Lt Col Bagley, whose enthusiasm and confidence in my abilities inspired me to try a road less travelled. He encouraged good research and independent thinking and his perspectives on "the bigger picture" were always appreciated. My thanks also to the other members of my committee, Maj Riggins and Capt Warhola for their valuable feedback. Thanks as well to Lt Mark Leonard, whose knowledge and enthusiasm provided a ready springboard for ideas.

A large support network had the daunting tasking of helping me maintain my sanity during the effort. As always, Mom and Dad were supportive through earfuls of triumphs and setbacks, and I love them for it. Jeff, Sherrill, Ken, Steve, and Bob gave me a proper introduction to the Institute, and I'll keep the end of tour tradition alive. Finally, my thanks to the members of the WNST&PG (Jeff, Joan, Pauleta, Neal, Kathleen, Sandi, and Jean-Luc, et al.) for splitting a sometimes unbearable week into bite-size chunks.

"But there are folks who want to know and aren't afraid to look and won't turn tail should they find it..." - Tom Robbins

- Jeffrey A. Blank

Table of Contents

List of Figures

AFIT/GA/ENY/91D-15

Abstract

Fractional calculus has been shown useful for describing many real world systems, and studies are currently underway to generalize control theory to incorporate fractional states. This investigation derives a method for simulating the time response of fractional order systems using a recursive difference equation. The technique used effectively approximates a simple fractional order integrator as a summation of integer order terms. The discrete transfer function is also derived and the frequency response of the discrete algorithm is compared to the exact continuous case. Using 20 or more retained past values in the difference equation, the discrete half-order integrator demonstrates a passband of more than three decades. A slightly modified method is used to derive a recursive difference equation which simulates the response of a modal fractional order differential equation. Frequency response analysis of an example having an eigenvalue of -1 shows the characteristics of a low pass filter, effectively simulating the continuous system response for all frequencies below the Nyquist limit, even when using a small number of retained past values.

Discrete Simulation of Fractional Order Systems

I. Introduction

I.1 Background

The key to understanding and controlling any system is to construct a model which accurately describes its behavior. Although integer order differential equations are most often used for such models, fractional order differential equations have been shown useful to describe some systems (10:1-3,197, 1:13-14). An example currently under study at AFIT involves viscoelastic material damping applicable to the design of large flexible space structures (3:304-306).

Integration and differentiation to fractional order was envisioned by L'Hospital in 1695 and defined in 1884 (12:115,118). It has seen limited use, however, since most systems can be approximately modelled to increasing accuracy by using increasingly large numbers of integer order terms. Enthusiasm in fractional order models stems from their ability to accurately model observed behavior for many systems with significantly fewer terms (1:13-14).

A relatively new application of fractional order models which is being explored is to control theory (7:92-94,3:311). Fractional order models may be useful, not only for systems exhibiting fractional order behavior, but also for systems well described by integer order models. Although not physically intuitive, fractional order states can be defined "in between" traditional integer order states, such as those corresponding to position, velocity, and acceleration. Use of these fractional order states in feedback control techniques are being investigated.

I.2 Motivation

Theoretical development and experimental work using fractional order models has generally dealt only with continuous or analog signals (2:4-6,7:92-94). Attempts to simulate fractional order systems in the laboratory using analog devices has met with limited success (7:92-94). This investigation seeks to lay the foundations for applying the power and flexibility of digital computers to the future study and application of fractional order system descriptions.

Simulating the response of a system to various outputs allows focused study in a laboratory environment. System responses to a variety of inputs can be easily simulated using a computer. System parameters may also be changed quickly and easily to allow study of a wide variety of cases. Digital computer simulation is dependent on a discrete representation of the continuous system model. Discrete models are also required for real-time system state estimation and control using digital computers. In this case, an effective algorithm for the discrete model must be relatively fast and efficient, requiring little computer memory and few calculations per time increment.

I.3 Objective

Discrete algorithms which simulate the response of simple fractional integrators will be derived. In addition, discrete algorithms will be derived which simulate the response of equations similar to fractional order differential equations. The algorithms should have as few terms as possible to minimize the computer memory and speed required. Since these discrete representations approximate continuous models, the frequency range of the input forcing function will be determined over which the algorithm output adequately simulates the exact response.

I.4 Scope

The continuous systems to be simulated are assumed to possess linear, constant coefficient models. The order of the fractional integrator or differentiator remains general throughout this investigation, but it is always assumed real. The development limits the system input to those functions for which Laplace transforms exist. Sinusoidal forcing functions are used to evaluate frequency response of the various discrete algorithms, but this should not be viewed as limiting the input forcing function to that class. Finally, the system is assumed in a rest state prior to the application of the input forcing function, so only zero-state initial conditions are considered.

I.5 Approach

The method used here to derive discrete algorithms for a continuous fractional order system first finds the sampled response of the system to a unit pulse input. This pulse response sequence, as it is called, is then used to calculate the coefficients of a recursive difference equation. This discrete algorithm uses past input and past output values to calculate the current output. Once the coefficients have been determined, this discrete algorithm simulates the time response of the continuous system for a wide range of input forcing functions.

Much of the rest of the effort is spent determining the conditions for which the difference equation adequately simulates the continuous response. For a given fractional order system and a given sampling period, the fidelity of the discrete simulation will depend only on the frequency content of the input forcing function, so the discrete algorithm is manipulated to achieve a discrete transfer function which is evaluated over a range of input frequencies. A frequency range of acceptable magnitude and phase response performance, called the passband, can be determined by comparing the discrete algorithm frequency response to the known continuous system frequency response over a wide range of input forcing function frequencies.

II. The Fractional Differintegral Operator

II.1 Definition

H. Laurent defined the fractional integration operator as (12:118)

$$\mathrm{I}^{\alpha}x(t) \equiv \mathrm{D}^{-\alpha}x(t) \equiv \frac{1}{\Gamma(\alpha)}\int_{c}^{t}\frac{x(t-\tau)}{\tau^{1-\alpha}}d\tau \qquad \alpha > 0 \tag{1}$$

where Γ is the gamma function or generalized factorial, given by (10:16)

$$\Gamma(x) \equiv \lim_{N\to\infty}\left[\frac{N!N^{x}}{x(x+1)(x+2)\cdots(x+N)}\right] \tag{2}$$

Eq (1) is usually called the Riemann-Liouville (R-L) integral, since it united two earlier competing definitions (12:118-119). Only functions for which x=0 for t<0 are considered, so the lower terminus of integration, c, will always be zero for this investigation.

Extending the R-L integral definition to fractional differentiation is not possible directly since the integral diverges for $\alpha \leq 0$. To solve this problem, integer order derivatives are used to pose a fractional derivative in terms of the R-L integral:

$$\mathrm{D}^{\alpha}x(t) \equiv \mathrm{D}^{\nu-\beta}x(t) \equiv \frac{\mathrm{d}^{\nu}}{\mathrm{dt}^{\nu}}\,\frac{1}{\Gamma(\beta)}\int_{0}^{t}\frac{x(t-\tau)}{\tau^{1-\beta}}d\tau \qquad -\infty < \alpha < +\infty \qquad \beta = \nu-\alpha \qquad 0 < \beta \leq 1 \tag{3}$$

where ν is the least nonnegative integer greater than α (12:119). Eq (3) is sometimes called the differintegral operator, since it incorporates integration and differentiation to any order into one equation (10:45).

II.2 Properties

The differintegral operator of Eq (3) has the following properties (12:118):

1) When α is a negative integer, the result is identical to ordinary α-fold integer order integration. Similarly, when α is a positive integer, the result is identical to ordinary α-fold integer order differentiation.

2) The operator is linear.

3) If the order of differintegration is zero, the operator leaves the function unchanged, i.e. $D^0[x(t)] = x(t)$.

4) The law of exponents holds, i.e. $D^m[D^n x(t)] = D^{m+n} x(t)$. This property, known as the composition law, can be seen in the derivation of the differintegral operator from the R-L integral above.

Note in the definition that the limits of the convolution integral are $\tau=0$ and $\tau=t$, where τ is the dummy variable of integration. Since the independent variable represents time in this investigation (without loss of generality), the definition implies knowledge of the entire past history of a function in order to calculate its differintegral. This presents no contradiction with intuitive notions of first order integration, which is often visualized as a summation of infintestimally thin areas to find the "area under the curve." The fractional order differintegral operator can be visualized as a weighted summation of these areas, in which the relative weighting value of each area is not constant as with integer order integration. This weighting function is dependent on the fractional order of integration and will be explored later.

II.3 Frequency Response

The discrete algorithms to be derived in this investigation will simulate the continuous time response of a fractional order system to a wide variety of time domain input forcing functions. As a tool for evaluation, all these possible input forcing functions can be represented by sinusoidal time functions of various frequencies. Since the fractional differintegral is a linear operator, the sinusoidal input will cause a sinusoidal output at the same frequency, which may have a different amplitude (magnitude) and/or phase compared to the input. The discrete algorithm's magnitude and phase response for a range of frequencies can then be used as performance measures by comparing them to the magnitude and phase response of the exact continuous case.

The continuous time response, y(t), of a simple fractional integrator system to an input function, u(t), is given by

$$y(t) = \mathrm{I}^{\alpha}u(t) = \frac{1}{\Gamma(\alpha)}\int_0^t \frac{u(t-\tau)}{\tau^{1-\alpha}}d\tau \qquad \alpha > 0 \tag{4}$$

The complex frequency response of this system can be found by performing a Laplace transformation of both sides:

$$\mathcal{L}\{y(t)\} = \mathcal{L}\left\{\frac{1}{\Gamma(\alpha)}\int_0^t \frac{u(t-\tau)}{\tau^{1-\alpha}}d\tau\right\} \tag{5}$$

The one-sided Laplace transform used here is defined by the following transform pair relations (11:143)

$$F(s) = \mathcal{L}\{f(t)\} \equiv \int_{0^-}^{\infty} f(\tau)\, e^{-s\tau}\, d\tau \tag{6}$$

$$f(t) = \mathcal{L}^{-1}\{F(s)\} \equiv \frac{1}{2\pi j}\int_{\sigma-j\infty}^{\sigma+j\infty} F(s)e^{st}ds \tag{7}$$

where $j=(-1)^{1/2}$ and s is a complex variable consisting of a real part, σ, and an imaginary part, $j\omega$. (Use of the one-sided transform is justified since u(t)=0 and y(t)=0 for t<0 in this investigation.) In order for these integrals to exist, the time domain function, f(t), must be sectionally continuous in every finite interval in the range t>0 and must be of exponential order as t approaches infinity (9:25). The input and response functions, u(t) and y(t), are assumed to meet these restrictions throughout this investigation in order to apply the Laplace transform technique. Most forcing functions of interest in engineering applications conform to this assumption, including the "sinusoidal" functions (turned on at t=0) which will be used to evaluate the discrete algorithms, so this does not greatly restrict the variety of possible system inputs.

Using the convolution property of the Laplace operator, Eq (5) becomes

$$Y(s) = \frac{1}{\Gamma(\alpha)} \mathcal{L}\{u(t)\} \mathcal{L}\{t^{\alpha-1}\} \tag{8}$$

where Y(s) is the Laplace transform of the output. The transform of the last term is (8:1022)

$$\mathcal{L}\{t^{\alpha-1}\} = \frac{\Gamma(\alpha)}{s^{\alpha}} \qquad \alpha > 0 \tag{9}$$

So the Laplace domain representation of a simple fractional integrator is

$$Y(s) = s^{-\alpha} U(s) \tag{10}$$

where U(s) is the Laplace transform of the input. The familiar rule for the Laplace transform of a simple integrator is seen to generalize to fractional order; the order of the integrator appears as a negative exponent of the Laplace variable, s.

The transfer function for the system is

$$H(s) = \frac{Y(s)}{U(s)} = s^{-\alpha} \tag{11}$$

The frequency response of the system can be found by evaluating the transfer function along the imaginary axis in the complex s-plane, that is $s=j\omega$. The resulting complex quantity, $H(j\omega)$, or $|H(j\omega)|e^{j\phi}$ in polar coordinates, represents the complex system response for any given input frequency ω. As mentioned, the frequency will not change from the input, but the magnitude and/or phase may. The relative magnitude of the output compared to the input is often plotted in decibels, where $dB=20\log_{10}(|H(j\omega)|)$, and the phase shift of the output compared to the input, ϕ, is often plotted in degrees. The input frequency, ω, will be plotted in radians per second.

Figure 1 shows the frequency response of a fractional integrator for several values of α. Given the extension of the Laplace transform for the simple integrator to fractional order, it is not surprising that the frequency response is simply an extension of the integer order case. The rule governing the magnitude plot for a simple integrator, slope=-20n dB/decade, where n is the order of integration, generalizes to fractional order. Similarly, the simple integrator phase response of a constant $\phi=-90n$ degrees generalizes to fractional order. These exact frequency responses can now be used to gauge the performance of discrete fractional integration algorithms.

II.4 Fractional Differentiator Frequency Response

The continuous time domain response, y(t), of a simple fractional differentiator system (limited to fractional order $0\leq\alpha<1$) to an input function, u(t), is

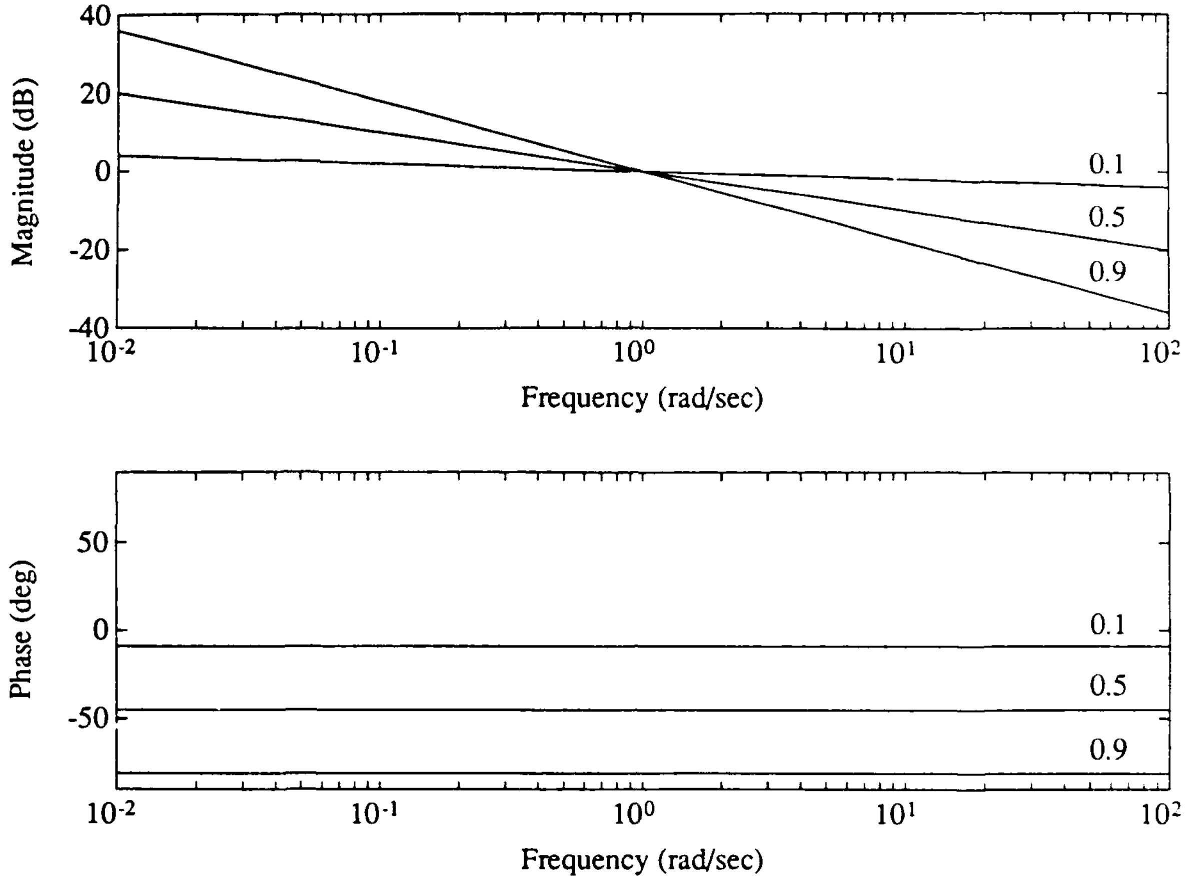


Figure 1 Fractional Integrator Exact Frequency Responses

given by

$$y(t) = \mathrm{D}^{\alpha}u(t) = \frac{1}{\Gamma(1-\alpha)}\,\frac{d}{dt}\int_0^t \frac{u(t-\tau)}{\tau^{\alpha}}d\tau \qquad (12)$$

$$0 \leq \alpha < 1$$

Prior to transforming this equation into the Laplace domain, Leibnitz' rule for differentiation of an integral is used to obtain

$$y(t) = \frac{1}{\Gamma(1-\alpha)} \int_0^t \frac{\dot{u}(t-\tau)}{\tau^\alpha} d\tau + \frac{u(0)}{t^\alpha \Gamma(1-\alpha)} \tag{13}$$

Taking the Laplace transform of both sides yields

$$Y(s) = \frac{1}{\Gamma(1-\alpha)} \mathcal{L}\{\dot{u}(t)\} \mathcal{L}\{t^{-\alpha}\} + \frac{u(0)}{\Gamma(1-\alpha)} \mathcal{L}\{t^{-\alpha}\} \tag{14}$$

Using the transform given in Eq (9) and the time differentiation property of the Laplace transform, $\mathcal{L}\{\dot{u}(t)\}=sU(s)-u(0^+)$, the equation becomes

$$Y(s) = [s\,U(s) - u(0)] \frac{1}{s^{1-\alpha}} + \frac{u(0)}{s^{1-\alpha}} \tag{15}$$

Since the last two terms cancel, the Laplace domain representation of a simple fractional order differentiator is

$$Y(s) = s^\alpha U(s) \tag{16}$$

This representation of a fractional order differentiator differs in one important aspect from that of integer-order differentiators; the initial condition of the input function does not appear (as in the time differentiation property used above). This has no bearing on the frequency response technique used here, but it does hint at complications to solution of the initial value problem for fractional differential operators (2:37-38). Otherwise, the Laplace transform of a simple differentiator generalizes to fractional order, with the order of differentiation appearing as a positive exponent on the Laplace variable, s.

The transfer function of the simple fractional differentiator is

$$\frac{Y(s)}{U(s)} = s^{\alpha} \tag{17}$$

Figure 2 shows that this operator also follows the frequency response rules of the integer order case. The magnitude slope is given by 20n dB/decade and the constant phase angle is given by 90n degrees, where n is the order of the differentiator.

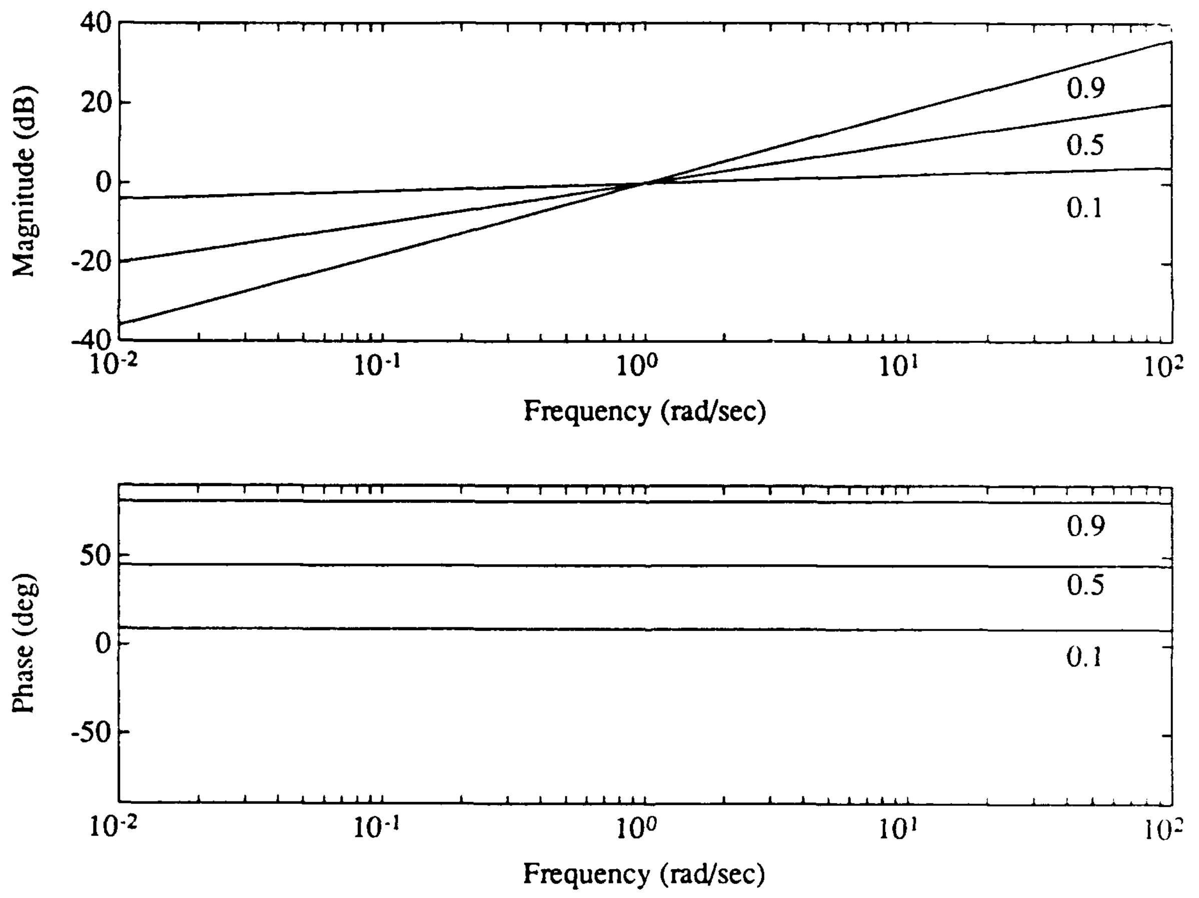


Figure 2 Fractional Differentiator Exact Frequency Responses

III. Integration - Convolution Summation Algorithm

The convolution summation will be shown to be a poor algorithm for discrete simulation of fractional order systems, since its frequency response does not closely match that of the continuous case. However, its simplicity allows straightforward explanation of many of the concepts which will be used throughout this investigation and yields valuable insight into the nature of the fractional integration operator.

III.1 Derivation

The derivation of a discrete algorithm for fractional integration begins with the continuous R-L integral:

$$y(t) = \mathrm{I}^{\alpha}u(t) = \frac{1}{\Gamma(\alpha)}\int_0^t \frac{u(t-\tau)}{\tau^{1-\alpha}}d\tau \qquad (18)$$

$$\alpha > 0$$

The first step is to make the substitution t=kT, k=1,2,..., and replace the integration operation with a summation of k integrals, each integrating over a small time interval, T, called the sample period. This yields

$$y(kT) = \frac{1}{\Gamma(\alpha)}\sum_{j=1}^{k}\int_{(j-1)T}^{jT}\frac{u(kT-\tau)}{\tau^{1-\alpha}}d\tau \qquad (19)$$

The output is said to be "sampled" and is identical to the continuous response only at integer multiples of the sample time, i.e.:

$$y(kT) \equiv y(t)\big|_{t=kT} \qquad k = 1,2,... \tag{20}$$

During the sample period, i.e. when $kT \le t < (k+1)T$, the sampled output, y(kT), holds a constant value while the continuous output may vary.

Next, the input forcing function is also sampled and approximated as constant over each sample period:

$$u(kT) \equiv u(t)\big|_{t=kT} \qquad k = 1,2,... \tag{21}$$

Here again, the sampled input, u(kT), is constant over the sample time, $kT \le t < (k+1)T$, while the actual input, u(t), may vary. This sample-and-hold (or piecewise-constant) approximation is a close approximation of the input signal for sufficiently small T, as will be discussed in more detail shortly. Since the input is now assumed constant over each sample period, it can be factored out of the integral:

$$y_T(kT) = \frac{1}{\Gamma(\alpha)} \sum_{j=1}^{k} \left[u((k-j)T) \int_{(j-1)T}^{jT} \tau^{\alpha-1} d\tau \right] \tag{22}$$

This new assumption changes the character of the output, highlighted by the addition of the T subscript to the output variable. In general, $y_T(kT)$ as defined above will not equal y(kT) as defined in Eq (19), since y(kT) represents the sampled output due to a continuous input, and $y_T(kT)$ represents the sampled output due to a sampled input. In other words, $y_T(kT)$ would equal y(kT) (and match the exact continuous response at multiples of the sample time), only in the unlikely case of a constant input or one changing solely at integer multiples of T (11:173).

The validity of the piecewise-constant input approximation of Eq (21) is governed by the Nyquist criterion. It stipulates that the sample period be less than half the period of the highest frequency component in the input signal in order for the sampled signal to unambiguously represent the input signal (5:81-82). Hence, the sample time must conform to

$$T < \frac{\pi}{\omega} \tag{23}$$

where ω is the highest input frequency component in rad/sec.

For any given value of the sample period, T, the piecewise-constant approximation is valid for input frequencies less than the Nyquist frequency (defined as π/T). For frequencies above the Nyquist frequency, the sample time is large relative to the quickly changing input. In this case, the sampled input function does not unambiguously represent the high frequency signal, but rather will be interpreted as the sampled input function of a lower frequency input. This phenomenon, wherein frequencies at or above the Nyquist frequency cause system responses corresponding to lower frequency inputs, is called folding or aliasing. In general, sampled-data systems are expected to be unreliable at input frequencies equal to or greater than the Nyquist frequency (5:81).

To complete the derivation, the integration is performed to yield

$$y_T(kT) = \frac{1}{\Gamma(\alpha)} \sum_{j=1}^{k} \left[u((k-j)T) \frac{(jT)^\alpha - ((j-1)T)^\alpha}{\alpha} \right] \tag{24}$$

$$\alpha \neq 0$$

Distributing the gamma function over the summation, the equation becomes

$$y_T(kT) = \sum_{j=1}^{k} u((k-j)T)\, h_T(jT) \tag{25}$$

where

$$h_T(jT) = \frac{(jT)^\alpha - ((j-1)T)^\alpha}{\Gamma(1+\alpha)} \tag{26}$$

Note that $(\alpha)\Gamma(\alpha)=\Gamma(1+\alpha)$. The convolution summation of Eq (25) represents the sampled response of a fractional integrator to a piecewise-constant input. This algorithm is similar to the R1 algorithm of Oldham and Spanier (10:138-139).

The problem with this convolution summation algorithm is that the number of past input values used in the calculation increases with each time step (value of k). This implies that for a given time step, k past input values and k fixed coefficients (h_T values) must be stored in memory (2k total). Additionally, 2k-1 calculations (k multiplications and k-1 additions must be performed to find the response value for that time step. Obviously, this representation is not reasonable for a real-time fractional integration algorithm which must function over any length of time.

Of course, one way to get around the storage and calculation problems inherent in the convolution summation is to limit the number of past values which will be retained and used in the calculation. If only m past input values are retained, Eq (25) becomes

$$y_T(kT) = \sum_{j=1}^{m} u((k-j)T)\, h_T(jT) \tag{27}$$

where only the range of the summation has changed. This formulation requires storage of 2m values, including m fixed coefficients (h_T values) and the m most recent input values, regardless of the value of k. The number of calculations required for each time step is also constant at 2m-1 (m multiplications and m-1 additions).

The convolution summation limited to m terms given in Eq (27) is an approximation of the exact convolution summation of Eq (25). For k values less than m, this formulation yields the same response as Eq (25), since u(iT)=0 for i<0. More important for the frequency response of the algorithm are those time steps greater than m. In that case, Eq (27) approximates the response of Eq (25) by summing only the m most recent values, weighted by the appropriate h_T value. This results in an inherent error for step function inputs (u(jT)=constant, j>0) and low frequency inputs where the period of the input is large compared to the sample time. As the number of past input values retained increases, the low frequency performance of the algorithm is expected to improve.

III.2 The Pulse Response Sequence

The h_T sequence calculated in Eq (26) is known as the pulse response sequence for the fractional integrator, since it is defined as the sampled response of the system to a unit pulse input. The unit pulse input itself is defined as (11:174)

$$\delta^*(t) \equiv \begin{Bmatrix} 1 & 0 < t \leq T \\ 0 & otherwise \end{Bmatrix} \tag{28}$$

The pulse response sequence is the key to deriving discrete system models. This sequence can be viewed as the weighting function discussed in the last chapter. (This is an inexact analogy, however, since the pulse response sequence contains the "width" of the rectangle, T, as well as the weighting factor.) It is interesting to note that the pulse response sequence can be relatively easily generated in many real world applications and applied to a system under study. The resulting sampled output is the pulse response sequence of that system (by definition) and can be used in the methods described in this investigation to derive system models regardless of any a priori knowledge of the underlying principles of the system (11:176).

The unit pulse can also be represented as the difference of two unit step functions, separated in time by the sample time duration:

$$\delta^*(t) = u_{-1}(t) - u_{-1}(t-T) \tag{29}$$

where $u_{-1}(t)$ indicates a unit step input. The continuous response of the fractional integrator to a unit step input is given by

$$r(t) = \frac{1}{\Gamma(\alpha)} \int_0^t \frac{1}{\tau^{1-\alpha}} d\tau \qquad \alpha > 0 \tag{30}$$

Performing the integration and sampling the output yields

$$r(kT) = \frac{(kT)^\alpha}{\Gamma(1+\alpha)} \tag{31}$$

Since the fractional integrator is a linear operator, the pulse response sequence can now be calculated as the difference of two of these unit step input responses, separated by one sample period:

$$h_T(kT) = r(kT) - r((k-1)T) = \frac{(kT)^\alpha - ((k-1)T)^\alpha}{\Gamma(1+\alpha)} \tag{32}$$

This is identical to the pulse response sequence definition given in Eq (26).

Fractional integrator pulse response sequences are given in Figure 3 for a sample time of 0.1 seconds (T=0.1) and various values of α. Past input values tend to be weighted equally for α near one. As α increases, more emphasis is placed on recent values compared to distant past values. In fact, for α=0.1, the pulse response sequence weights the most recent input much more heavily than past inputs. This corresponds to a weighting sequence approaching a zero order integrator, which passes the current input forcing function directly to the output.

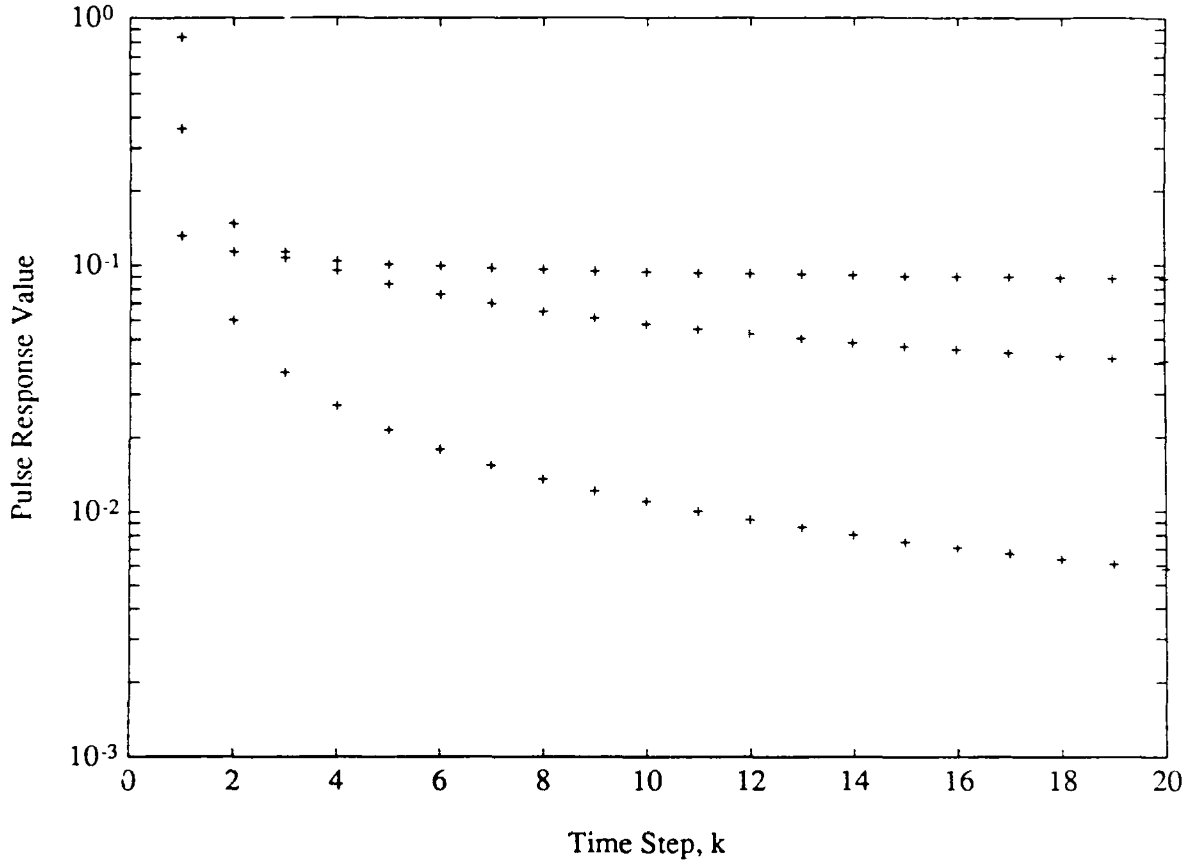


Figure 3 Fractional Integrator Pulse Response Sequences

III.3 Pole-Zero Structure

The convolution summation algorithm simulates the time response of a continuous fractional integrator. As discussed earlier, an effective way of measuring the fidelity of the discrete algorithm is to compare its magnitude and phase frequency responses to that of the continuous fractional integrator over a wide range of input forcing function frequencies. It is then possible to specify a band of input frequencies over which the discrete algorithm simulates the continuous system response to the accuracy required for a given application.

In order to directly compare the frequency responses of the discrete and the continuous representations, a new operator must be defined. The one-sided z-transform of a discrete sequence f(k), k=0,1,2,..., is defined as

$$Z\{f(k)\} \equiv F(z) \equiv \sum_{k=0}^{\infty} f(k) z^{-k} \tag{33}$$

provided the summation converges (11:196). This is a discrete analogue of the Laplace transformation. The complex variable, z, can also be defined in terms of the Laplace variable and the sample period using $z \equiv e^{sT}$.

The convolution summation of Eq (25) can also be written as

$$y_T(kT) = \sum_{j=1}^{k} u(jT)\, h_T((k-j)T) \tag{34}$$

Taking the z-transform of both sides yields

$$Z\{y_T(kT)\} = Y(z) = Z\left\{ \sum_{j=1}^{k} u(jT)\, h_T((k-j)T) \right\} \tag{35}$$

where Y(z) is the z-transform of the sampled output. Analogous to the Laplace transform of a convolution integral, the z-transform of a convolution summation is equal to the product of the individual z-transforms (6:126). Hence, Eq (35) can be expressed as

where U(z) is the z-transform of the sampled input function. Applying the

$$Y(z) = Z\{u(kT)\}\, Z\{h_T(kT)\} = U(z)\, Z\{h_T(kT)\} \tag{36}$$

definition of the z-transform yields the exact discrete transfer function for the convolution summation:

$$H(z) = \frac{Y(z)}{U(z)} = \sum_{j=0}^{k} h_T(jT) z^{-j} \tag{37}$$

This formulation reiterates the fact that the number of terms in the algorithm increases with each time step, since the transfer function changes based on the value of k.

Limiting the series to m terms and realizing that $h_T(0)=0$ yields the transfer function corresponding to the approximate convolution summation of Eq (27):

$$H(z) = \sum_{j=1}^{m} h_T(jT) z^{-j} \tag{38}$$

Multiplying the numerator and denominator by z^m and rearranging another form of the discrete transfer function:

$$H(z) = \frac{Y(z)}{U(z)} = \frac{\sum_{k=1}^{m} h_T(kT) z^{m-k}}{z^m} \tag{39}$$

This discrete transfer function will be used to find the frequency response of this discrete simulation of the continuous fractional integrator.

Although not required to find the frequency response, the discrete transfer function can be factored to yield

$$H(z) = \frac{G \sum_{j=1}^{m-1} (z - z_j)}{\sum_{i=1}^{m} (z - p_i)} \tag{40}$$

G is the transfer function gain (which is equal to $h_T(1T)$ in this case), z_j is a discrete zero, and p_i is a discrete pole. The discrete zeros and poles can be plotted on the complex z-plane. The discrete pole-zero structure in the complex z-plane indicates characteristics of the discrete system response in much the same way as the continuous pole-zero structure in the complex s-plane.

The z-plane pole-zero structure of a typical convolution summation fractional integration algorithm is given in Figure 4, where "o" indicates a zero and "x" indicates a pole. This example uses 8 retained terms (m=8) and a sample time of 0.1 seconds (T=0.1) to simulate a half order integrator (α=0.5). Interestingly, this roughly circular pattern of the z-plane zeros always occurs for the convolution summation approximation of the half order integrator regardless of the number of retained past values, m. There is an mth order repeated pole located at the origin. The unit circle is also indicated on the z-plane plot; the complex z-plane enclosed by the unit circle corresponds to the left-half s-plane as the stable region for pole locations.

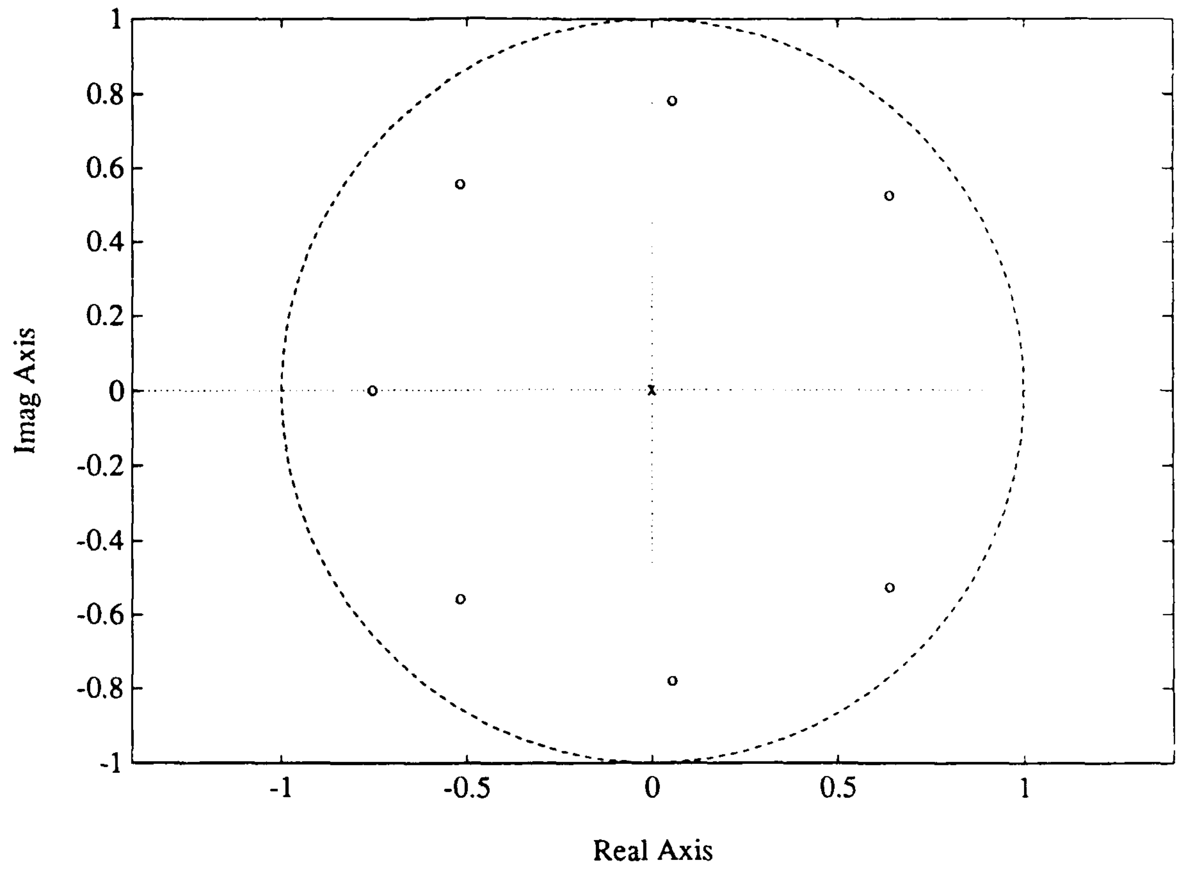


Figure 4 Convolution Summation z-Plane Pole-Zero Structure

III.4 Frequency Response

With the discrete transfer function of Eq (39), the response of the convolution summation algorithm to a range of input forcing function frequencies can be found and compared to the exact continuous response. Since the transfer function is in the discrete z-domain, it must be evaluated at z-plane points which correspond to the s-plane imaginary axis. These z-plane points form a unit radius circle surrounding the origin, and can be calculated from s-plane imaginary axis points using the relationship $z=e^{j\omega T}$. (This unit circle is included in all z-plane plots in this investigation.) Evaluating the transfer function at various complex values of $z=e^{j\omega T}$ yields the relative magnitude $|H(e^{j\omega T})|$ and phase shift, ϕ, of the output of the convolution summation for various input forcing function frequencies.

Figure 5 shows the frequency response of the same half-order convolution summation example discussed above and the frequency response of the exact continuous half-order integrator. The magnitude response is surprisingly good for

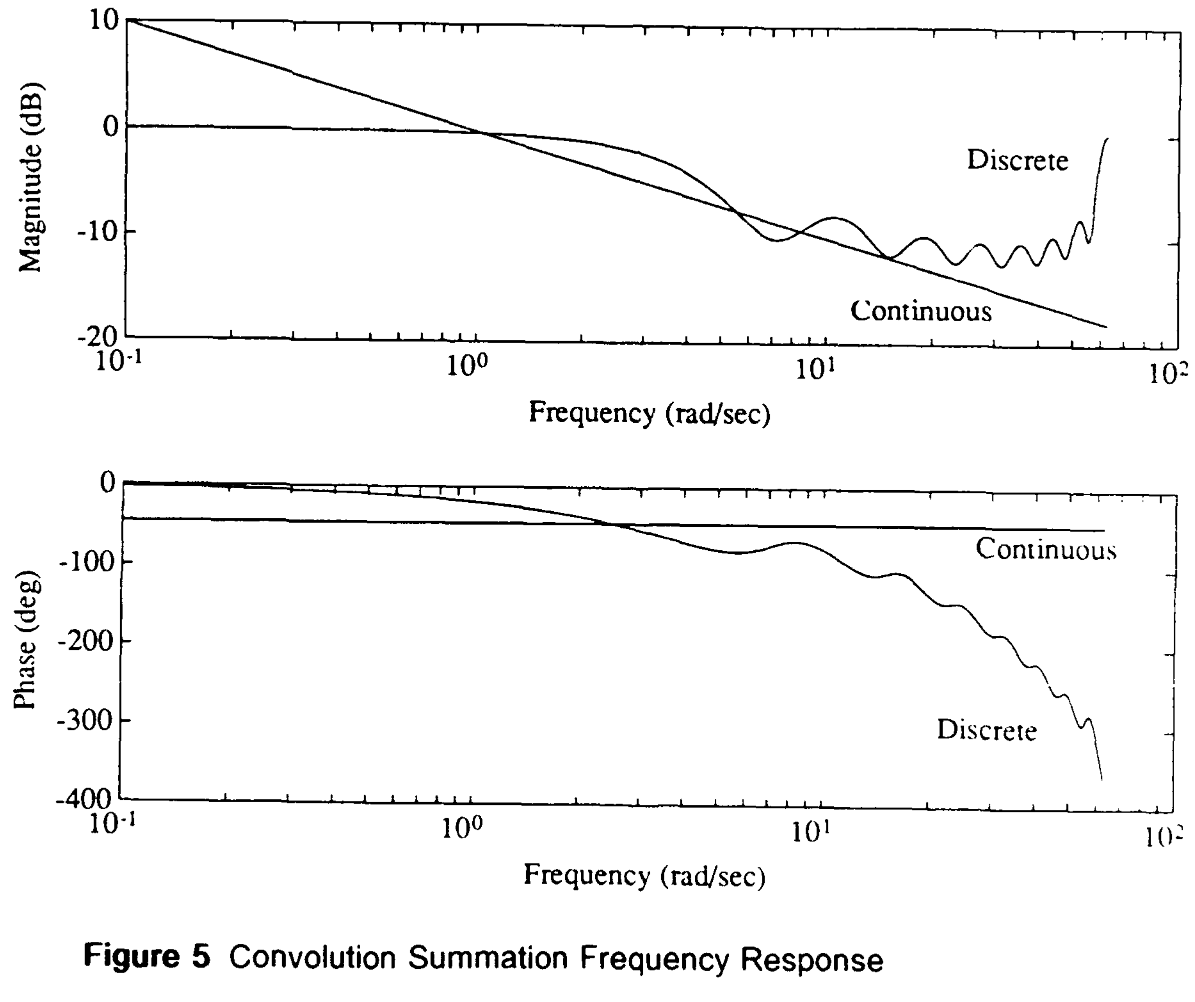


Figure 5 Convolution Summation Frequency Response

this simple algorithm, roughly simulating the exact continuous magnitude response for nearly two decades. Since T=0.1 sec in this case, the Nyquist frequency is $\pi/T=31.4$ rad/sec. As expected, input forcing function frequencies higher than the Nyquist frequency result in poor discrete simulation of the exact continuous case.

This algorithm obviously does a poor job of simulating the continuous phase response. Although not shown here, the phase response of the discrete simulation is not improved by increasing the number of past values retained. This deficiency makes the convolution summation useless for most system simulation and control applications.

This chapter has introduced the concepts which will be used to derive and analyze the discrete algorithms in the remainder of this investigation. The next chapter shows that a recursive difference algorithm is able to discretely simulate the magnitude and phase response of a continuous fractional integrator over a wide frequency range.

IV. Integration - Difference Equation Algorithm

IV.1 Derivation

The differintegral operator has been called a fading memory operator, since the current output is a function of all previous inputs, weighted to favor the recent past. Chapter III used this concept to derive a discrete convolution summation algorithm using only a weighted summation of a fixed number of previous input values to calculate the current output. However, this algorithm was shown inadequate for simulating a continuous simple fractional integrator.

The differintegral operator is also a hereditary operator, meaning that the current output depends on the state of the system (the previous response value, in this case) as well as the input. This is consistent with the concept of a weighted summation of all previous inputs, since the previous outputs are simply functions of previous inputs. In fact, algorithms can be derived for discrete simulation of the continuous fractional integrator using various combinations of past input and response values to calculate the current response.

A discrete algorithm for fractional integration which uses an equal number of past input and past response values is the finite difference equation:

$$\begin{aligned} y_T(kT) = &- a_{n-1}y_T((k-1)T) - a_{n-2}y_T((k-2)T) - \cdots - a_0 y_T((k-n)T) \\ &+ b_{n-1}u((k-1)T) + b_{n-2}u((k-2)T) + \cdots + b_0 u((k-n)T) \end{aligned} \tag{41}$$

Since the current output value depends on past values of the output, this is called a recursive equation. The total number of retained terms in this representation is 2n; n past input values and n past output or response values. This algorithm requires storage of the 2n coefficients and 2n past values, while requiring 4n-1 calculations for every time step (2n multiplications and 2n-1 additions).

The same approximations discussed for the convolution summation also apply here. Since the input signal is sampled, the high end frequency response will be limited based on the Nyquist criterion. Also, low frequency performance will be degraded since this algorithm uses a finite number of terms to represent what is, in effect, an infinite summation.

The following method for calculating the coefficients of the difference equation is paraphrased from Reid's illustration of the process (11:173-178). This method makes use of the pulse response sequence, so it is assumed known, either experimentally or calculated using Eq (26) using a given order of integration and sample time. Once the pulse response sequence is known, the coefficients of Eq (41) can be uniquely determined. The coefficients will vary depending on the value of n, the number of past input and response values retained, so this must be decided prior to calculating the coefficients.

From the definition given by Eq (28), a unit pulse input to the differenc equation requires u(0T)=1 and u(iT)=0 for i>0. By definition, the output at successive time steps will then be the pulse response sequence, i.e. $y_T(kT)=h_T(kT)$. Making these substitutions into Eq (41) and propogating the response for the first 2n time steps yields

$$\begin{aligned}
h_T(1T) &= b_{n-1} \\
h_T(2T) &= -a_{n-1}h_T(1T) + b_{n-2} \\
h_T(3T) &= -a_{n-1}h_T(2T) - a_{n-2}h_T(1T) + b_{n-3} \\
&\vdots \\
h_T(nT) &= -a_{n-1}h_T((n-1)T) - \cdots - a_1h_T(1T) + b_0 \\
h_T((n+1)T) &= -a_{n-1}h_T(nT) - \cdots - a_1h_T(2T) - a_0h_T(1T) \\
&\vdots \\
h_T((2n)T) &= -a_{n-1}h_T((2n-1)T) - \cdots - a_1h_T((n+1)T) - a_0h_T(nT)
\end{aligned} \tag{42}$$

Since the pulse response sequence is known, the system of equations in Eq (42) contains 2n equations and 2n unknowns, and can be solved to yield the a and b coefficients.

The a_i coefficients are calculated first by reformulating the last n equations of Eq (42):

$$\begin{bmatrix} h_T(1T) & h_T(2T) & \cdots & h_T(nT) \\ h_T(2T) & h_T(3T) & \cdots & h_T((n+1)T) \\ \vdots & \vdots & & \vdots \\ h_T(nT) & h_T((n+1)T) & \cdots & h_T((2n-1)T) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix} = \begin{bmatrix} -h_T((n+1)T) \\ -h_T((n+2)T) \\ \vdots \\ -h_T(2nT) \end{bmatrix} \tag{43}$$

This equation can be solved by using a variety of linear algebra methods. The n x n matrix on the left side of Eq (43) is called the Hankel matrix. The b_i coefficients can then be found by rewriting the first n equations of Eq (42):

$$\begin{aligned}
b_{n-1} &= h_T(1T) \\
b_{n-2} &= h_T(2T) + a_{n-1}h_T(1T) \\
b_{n-3} &= h_T(3T) + a_{n-1}h_T(2T) + a_{n-2}h_T(1T) \\
&\vdots \\
b_0 &= h_T(nT) + a_{n-1}h_T((n-1)T) + \cdots + a_1h_T(1T)
\end{aligned} \tag{44}$$

With the coefficients determined, the recursive difference algorithm of Eq (41) is a discrete time simulation of a continuous fractional integrator.

IV.2 Pole-Zero Structure

Using the time delay property of the z operator, Eq (41) becomes

$$\begin{aligned} Y(z) = &- a_{n-1}z^{-1}Y(z) - a_{n-2}z^{-2}Y(z) - \cdots - a_0z^{-n}Y(z) \\ &+ b_{n-1}z^{-1}U(z) + b_{n-2}z^{-2}U(z) - \cdots + b_0z^{-n}U(z) \end{aligned} \tag{45}$$

This yields a discrete transfer function of

$$H(z) = \frac{b_{n-1}z^{n-1} + b_{n-2}z^{n-2} + \cdots + b_0}{z^n + a_{n-1}z^{n-1} + a_{n-2}z^{n-2} + \cdots + a_0} \tag{46}$$

Similar to the convolution summation, this discrete transfer function can now be factored into discrete poles and zeros, and plotted on the complex z-plane.

Figure 6 gives the z-plane pole-zero strucure for a typical recursive difference algorithm. This example involves simulating the same half order integrator originally discussed in the convolution summation example. The sample time remains 0.1 seconds and the total number of retained terms remains 8, although this now consists of 4 past input values and 4 past output values. The poles and zeros are distinct and line up on the positive real axis of the z-plane.

As mentioned in Chapter III, the discrete pole-zero structure in the z-plane offers the same information available with the continuous pole-zero structure in

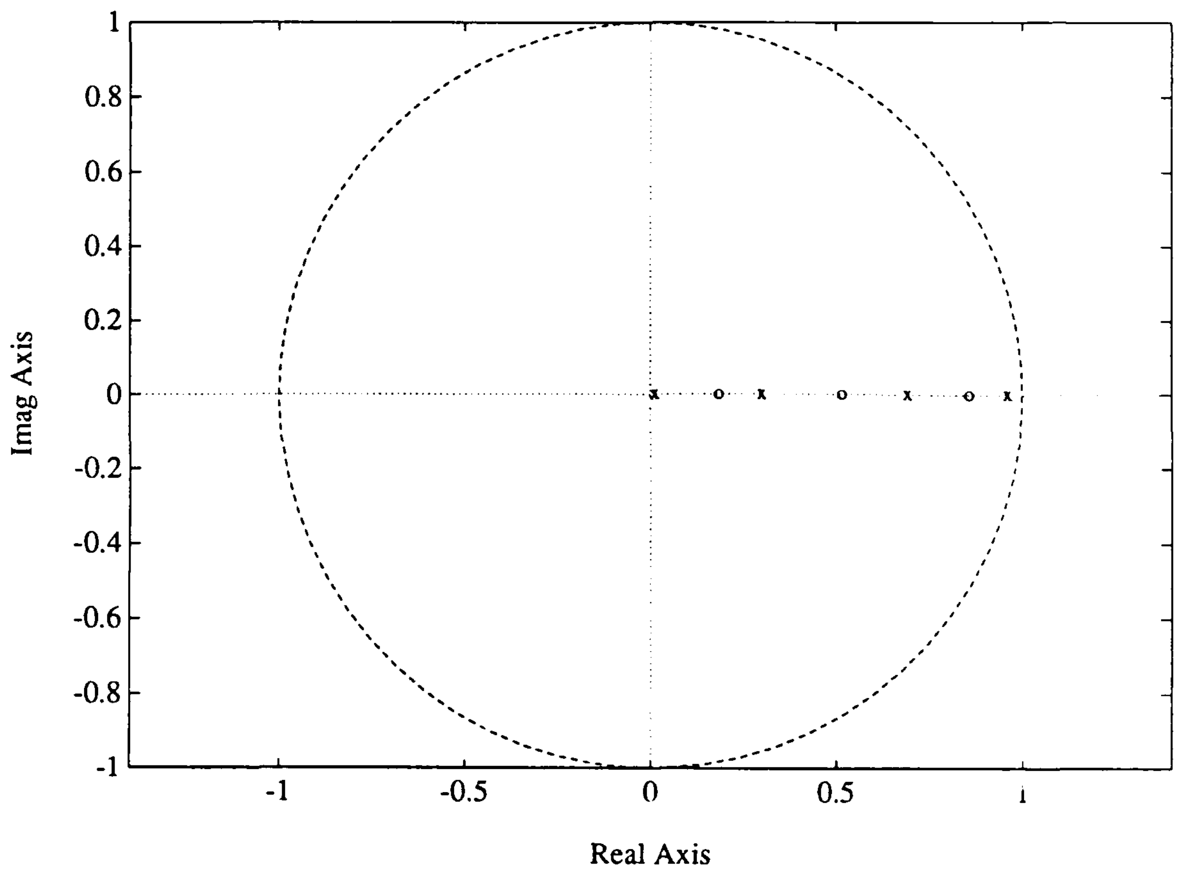


Figure 6 Integrator Example z-Plane Pole-Zero Structure

the s-plane. However, the z-plane is usually not as familiar as the Laplace domain to most engineers. Transforming the discrete poles zeros into the continuous domain yields some additional insight into the discrete simulation of the fractional integrator.

Mapping the poles and zeros of the discrete fractional integration algorithm from the z- to the s-plane will be accomplished using the Tustin transformation:

$$z = \frac{2 + sT}{2 - sT} \tag{47}$$

This is a bilinear transformation, meaning that the number of poles and zeros will be equal in the new plane regardless of unequal numbers in the old plane. This transformation can be derived by approximating $z=e^{sT}$ as a finite series. and it can be shown to be very accurate up to the Nyquist frequency for a given sample time (6:246). This transformation to allow continuous frequency response comparison of discrete systems is sometimes called a w-transformation (6:216-217).

The resulting continuous poles and zeros represent a continuous integer order transfer function which has the same frequency response (in the range of interest) as the continuous fractional integrator. In fact, to the accuracy of the Tustin transformation, the continuous integer order transfer function will possess a pulse response sequence identical to that of the fractional integrator. Hence, they share the same recursive difference equation representation.

Using the Tustin transformation, this discrete fractional integrator example yields the continuous s-plane pole-zero structure given in Figure 7. Note that all but one of the poles and zeros line up alternately on the negative real axis. The exception is the pole at s=20 (=2/T), which was "created" by the bilinear transformation as a result of the sample-and-hold operation on the input. This produces a non-minimum phase system, indicating that poor transient time domain response can be expected for the sampled discrete system (5:33-34).

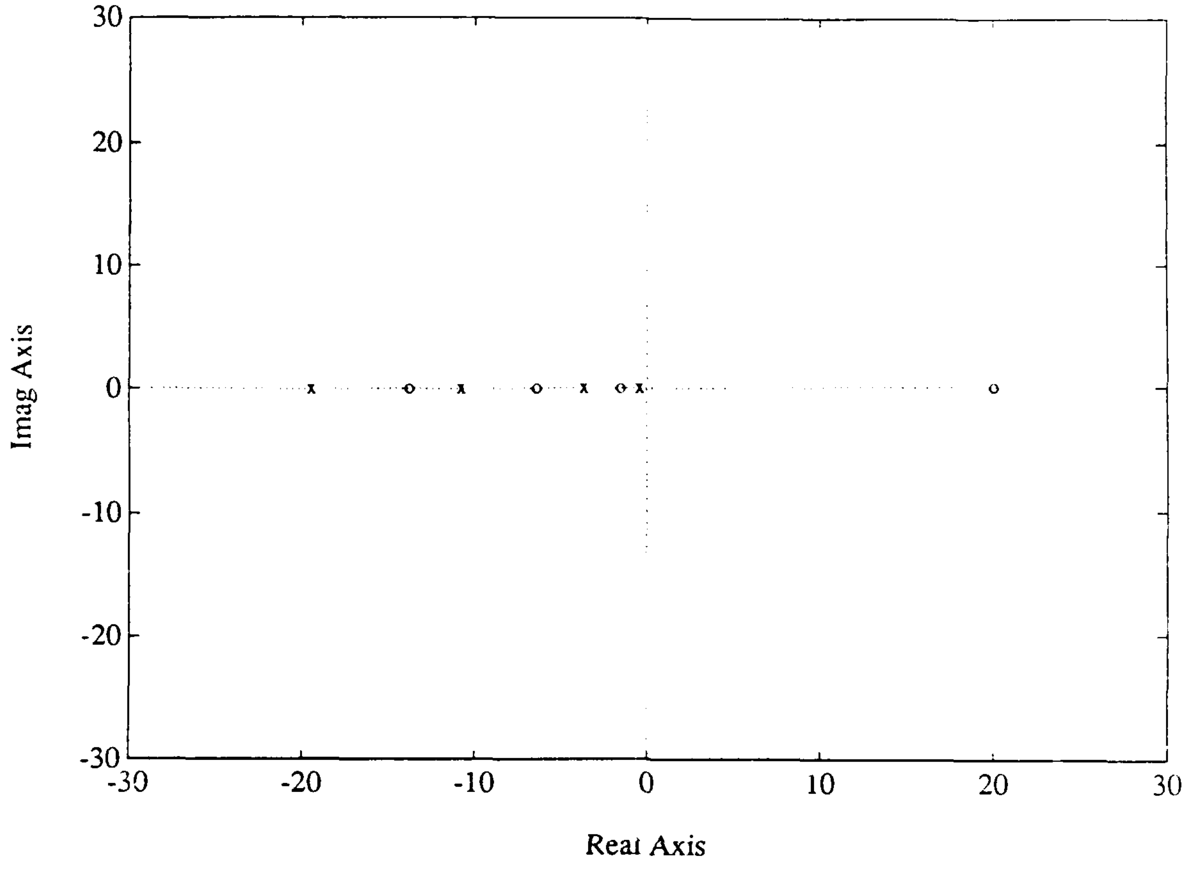


Figure 7 Integrator Example s-Plane Pole-Zero Structure

The poles and zeros in the s-plane can be directly related to the algorithm frequency response plot which will be given in the next section. Focusing for the moment on the magnitude response, each of these terms tend to contribute an additive ±20 dB/decade slope at frequencies higher than their breakpoint, which is the distance of the pole or zero from the origin along the real axis. By appropriately spacing alternate poles and zeros, the magnitude plot smoothly approximates the -10 dB/decade slope desired with a half order integrator. The residues of the s-plane poles and zeros can also be directly related to the time response of the system.

Since pole and zero locations define the frequency response breakpoints, they will also roughly define the range of acceptable performance of the algorithm. Poles and zeros closer to the origin will accompany better low frequency performance for the algorithm, and conversely, poles and zeros further from the origin will accompany better high frequency performance.

IV.3 Frequency Response

Using the method described for the convolution summation, the recursive difference algorithm can be evaluated to determine its frequency response. Figure 8 shows the frequency response of this algorithm and the exact continuous case for the half-order integrator whose pole-zero structures for both planes are given above. The phase response for this algorithm has the same general shape as that of the convolution summation, but can now be seen to approximate the continuous case over a range of frequencies roughly the same as the magnitude response passband. The magnitude response itself has improved, closely approximating the continuous case for over 2 decades. Once again, the upper end of the useful range is near the Nyquist frequency, with aliasing problems at higher frequencies. This algorithm is a marked improvement over the convolution summation and would be useful for simulating the response of a fractional integrator over a wide frequency range.

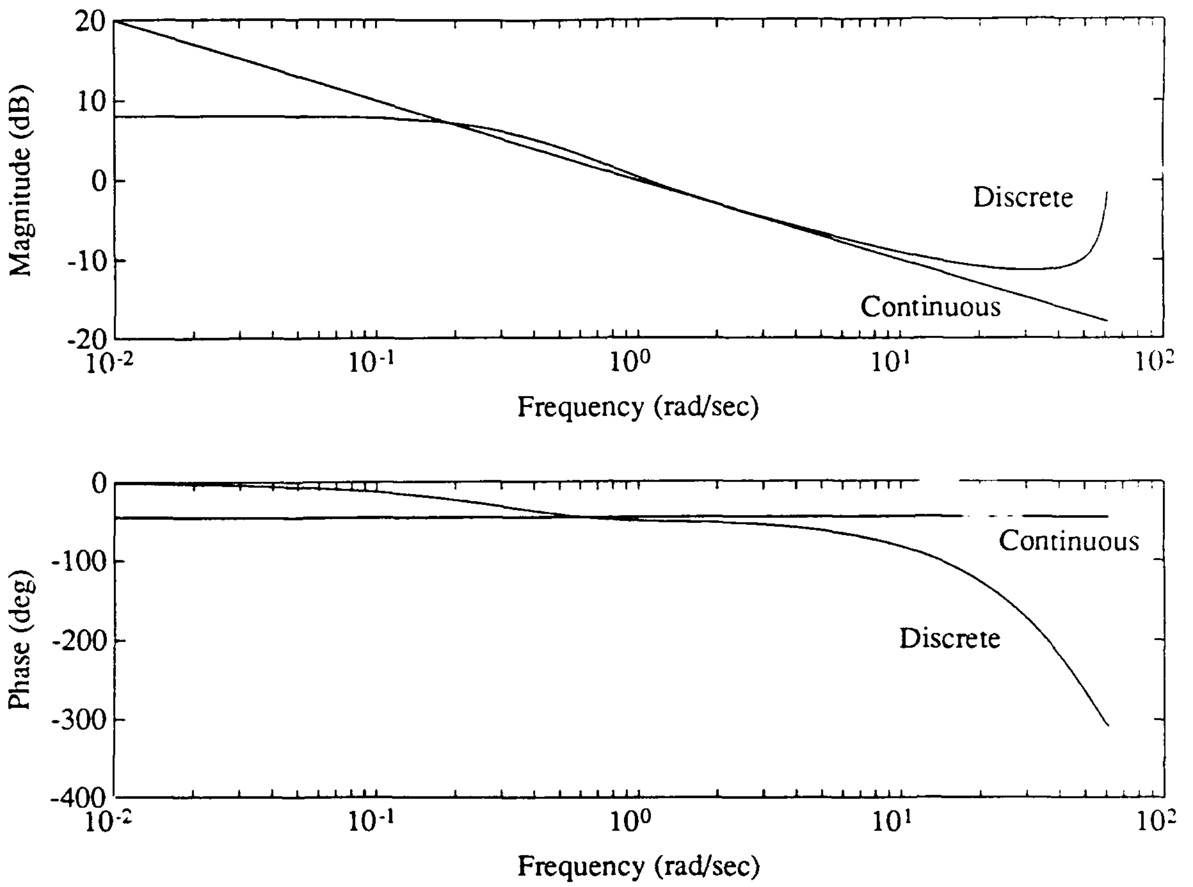


Figure 8 Integrator Example Frequency Response

Appendix B shows several time response plots using this discrete fractional integration algorithm for several input forcing function frequencies: $\pi/4$, $\pi/2$, and π (all in rad/sec). The magnitude and phase response of the discrete algorithm read from these time response plots confirms the magnitude and phase frequency response plots for those specific frequencies. The time response plots show that the recursive difference equation does simulate the response of a continuous fractional integrator.

In order to compare the performance of the recursive finite difference algorithm for a variety of cases, criteria can be established to define the algorithm passband in terms of the absolute magnitude and phase error with respect to the exact continuous case. For this study, the passband is defined as the largest contiguous frequency range over which the magnitude error is less than 3 dB and the phase error is less than 60 degrees. Using these criteria, a large number of different magnitude and phase plots like Figure 8 can be compared to bring out trends in the performance of the algorithm. The passbands obtained will be valid only for the criteria defined above.

Figure 9 gives the passband of the recursive difference algorithm simulating a half order integral for several values of sample time, T, and number of retained values, n. As expected, increasing the number of retained values increases the performance of the algorithm at the low frequencies. The upper passband limit is dependent on the sample time through the Nyquist criterion.

Another characteristic of the passband plots that is not so readily apparent is that the bandwidth (the "width" of the passband measured in decades of the frequency spectrum), is not dependent on the sample time. While decreasing the sample time improves the high frequency performance, it also degrades the low frequency performance by the same degree. Therefore, it is possible to "place" the bandwidth of the algorithm anywhere along the frequency spectrum by varying the sample time.

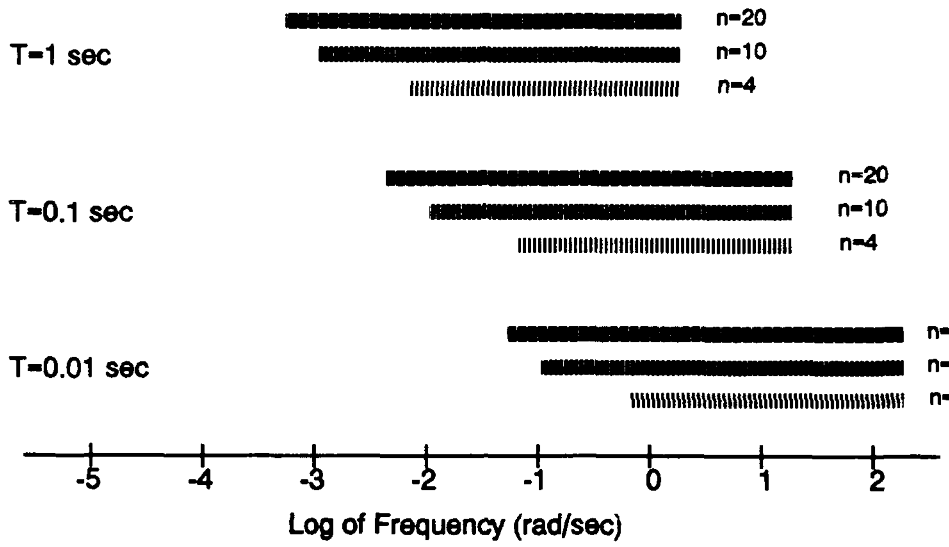


Figure 9 Integrator Example Passbands

This chapter has shown that the response of a continuous fractional integrator system can be successfully simulated over a wide frequency range using the discrete recursive difference algorithm. Chapter V will show that a pulse response sequence shows singularities for the simple fractional differentiator, so this technique is limited to fractional integrators (or as Chapter VI will show, those systems which can be reformulated as fractional integrators).

V. Differentiation

The pulse response sequence has been shown to be an important tool for deriving discrete algorithms to simulate the simple fractional integrator. Unfortunately, this technique does not generalize to the simple fractional differentiator. This brief chapter will show that the simple fractional differentiator has an unbounded pulse response sequence, which makes derivation of a discrete algorithm impossible using the method described in Chapter IV.

The continuous response of a simple fractional differentiator system to a input, u(t), is given by

$$y(t) = \mathrm{D}^{\alpha}u(t) = \frac{1}{\Gamma(1-\alpha)}\frac{d}{dt}\int_0^t \frac{u(t-\tau)}{\tau^{\alpha}}\,d\tau \qquad (48)$$
$$0 \leq \alpha < 1$$

To derive the pulse response sequence, the unit pulse will be represented here by the difference of two step responses, separated by one sample period. (This was discussed in Chapter III.) For a unit step input, u(t)=1 for t>0, so the system's continuous unit step response, r(t), is

$$r(t) = \frac{1}{\Gamma(1-\alpha)}\frac{d}{dt}\int_0^t \tau^{-\alpha}d\tau = \frac{t^{-\alpha}}{\Gamma(1-\alpha)} \qquad (49)$$

The continuous pulse response can now be found by subtracting a delayed step response from this one, yielding

$$h_T(t) = r(t) - r(t-T) = \frac{t^{-\alpha}}{\Gamma(1-\alpha)} - \frac{(t-T)^{-\alpha}}{\Gamma(1-\alpha)} \tag{50}$$

As before, the time scale is now partitioned into k equal time steps of T duration. This is the same as saying the output has been sampled, so the sampled pulse response sequence is

$$h_T(kT) = \frac{(kT)^{-\alpha} - ((k-1)T)^{-\alpha}}{\Gamma(1-\alpha)} \tag{51}$$

Note that this equation for the pulse response sequence of a fractional differentiator (limited to orders $0 \leq \alpha < 1$) is identical that for a fractional integrator given by Eq (26). Once again, this points back to the inverse relationship between the two operators.

The problem comes about when pulse response sequence values need to be calculated using Eq (51). For k=0, the definition yields $h_T(0T)=+\infty$ and for k=1, the definition yields $h_T(1T)=-\infty$. Since a simple differentiator of any order responds to instantaneous changes in the input, this result makes sense for a pulse input with infinite positive slope at k=0 and infinite negative slope at k=1. Obviously, these unbounded pulse response sequence values for k=0 and k=1 yield unbounded values in the Hankel matrix and make calculation of the difference equation coefficients impossible.

So it has been shown that the pulse response sequence method of deriving a discrete fractional differentiation algorithm will not work. This apparent gap in the types of systems which can be discretely simulated should not be of great concern, however. Simple differentiators of any order are avoided in practice since they are susceptible to noise. Where possible, systems requiring a differentiator should be recast in terms of fractional integrators. This technique will be used successfully in the next chapter to represent a fractional order differential equation as a fractional integrator system allowing derivation of a discrete algorithm for simulating its response.

VI. Fractional Differential Equations

VI.1 Definition

The preceding chapters developed algorithms for discrete simulation of a simple fractional integrator. The method can be extended to simulation of larger, more complex systems involving many terms and different orders of integration and/or differentiation. Systems are most often described using differential operators, so this chapter will focus on fractional order differential equations (FDE's).

The system under study is assumed to have been decomposed into a number of uncoupled modal fractional differential equations, each of the form

$$D^{\alpha} y_i(t) - \lambda_i y_i(t) = u_i(t) \qquad 0 \le \alpha < 1 \tag{52}$$

Bagley shows this process for a general constant coefficient system incorporating fractional derivatives (2:7-12). This is similar to decomposition of coupled integer order differential equations into a number of decoupled first order differential equations. The difference is that the original model may now contain fractional order terms and the resulting decoupled equations are, in general, of fractional order.

It is important to realize that Eq (52) is valid as a fractional differential equation only for the special case of a zero initial condition. True fractional differential equations must be posed in terms of a slightly different operator in order to accommodate initial conditions and to possess other than a trivial homogeneous solution (2:13-18). Such a restriction does not affect this investigation, since initial conditions are ignored for frequency response analyses anyway.

VI.2 Continuous Time Solution

Solving for the continuous response of Eq (52) begins with a Laplace transformation:

$$s^{\alpha}Y(s) - \lambda Y(s) = U(s) \tag{53}$$

Note that the "i" subscript denoting the specific modal equation has been dropped. The Laplace domain equation can be rearranged to yield

$$Y(s) = \frac{U(s)}{s^{\alpha} - \lambda} \tag{54}$$

Since multiplication in the Laplace domain is equivalent to convolution in the time domain, the time domain FDE solution is

$$y(t) = \int_{0}^{t} \mathcal{L}^{-1}\left\{\frac{1}{s^{\alpha}-\lambda}\right\} u(t-\tau)d\tau \tag{55}$$

where '$\mathcal{L}^{-1}$' indicates the inverse Laplace transform operation.

A power series expansion of the function can be used to find the inverse

Laplace transform:

$$\mathcal{L}^{-1}\left\{\frac{1}{s^{\alpha}-\lambda}\right\} = \mathcal{L}^{-1}\left\{\frac{1}{s^{\alpha}} + \frac{\lambda}{s^{2\alpha}} + \frac{\lambda^2}{s^{3\alpha}} + \frac{\lambda^3}{s^{4\alpha}} + \dots\right\} \tag{56}$$

This series converges for bounded values of s which satisfy $|s|>\lambda^{1/\alpha}$. Distributing the operator over the summation and using the transform relationship from Eq (9) for each of the terms yields

$$\mathcal{L}^{-1}\left\{\frac{1}{s^{\alpha}-\lambda}\right\} = \frac{t^{\alpha-1}}{\Gamma(\alpha)} + \frac{\lambda t^{2\alpha-1}}{\Gamma(2\alpha)} + \frac{\lambda^2 t^{3\alpha-1}}{\Gamma(3\alpha)} + \frac{\lambda^3 t^{4\alpha-1}}{\Gamma(4\alpha)} + \dots \tag{57}$$

which is valid for all t. In summation form, this is

$$\mathcal{L}^{-1}\left\{\frac{1}{s^{\alpha}-\lambda}\right\} = \sum_{p=1}^{\infty}\frac{\lambda^{p-1}t^{p\alpha-1}}{\Gamma(p\alpha)} \tag{58}$$

With the inverse Laplace transform in hand, the solution to Eq (52) is

$$y(t) = \int_0^t \left(\sum_{p=1}^{\infty}\frac{\lambda^{p-1}\tau^{p\alpha-1}}{\Gamma(p\alpha)}\right)u(t-\tau)\,d\tau \tag{59}$$

The time-domain solution can be presented in a more compact form by recognizing the alpha order Mittag-Leffler function (2:19):

$$E_{\alpha}(x) = \sum_{p=0}^{\infty}\frac{x^p}{\Gamma(1+p\alpha)} \tag{60}$$

This function can be viewed as a generalized exponential; note that it is the power series representation of an exponential for $\alpha=1$. Using Mittag-Leffler function notation, the FDE solution is (2:19-20)

$$y(t) = \int_0^t D^{1-\alpha}[E_\alpha(\lambda t^\alpha)]u(t-\tau)d\tau \tag{61}$$

VI.3 Frequency Response

The transfer function of the system is obvious from the Laplace domain representation of Eq (53):

$$H(s) = \frac{Y(s)}{U(s)} = \frac{1}{s^\alpha - \lambda} \tag{62}$$

Figure 10 shows the frequency response for this system for λ=-1 and several values of α. For low frequencies relative to λ, which is the breakpoint, the transfer function is approximately 1, so the magnitude and phase for all cases tend toward zero. The breakpoint at 1 rad/sec (equal to λ) is clearly seen in the responses of the α=0.9 case, which approaches the response of a first order term. The breakpoint is less obvious for the α=0.5 case and practically imperceptible for α=0.1. As α decreases, the frequency range over which $s^\alpha \approx 1$ increases, stretching out the transition region. For higher frequencies, each case approaches the response of a pure integrator.

VI.4 Pulse Response Sequence

Starting with the continuous time solution of Eq (59), the now familiar method of discretizing the integral and assuming a piece-wise constant input can be used to yield

$$y_T(kT) = \sum_{j=1}^{k}\left[u((k-j)T)\int_{(j-1)T}^{jT}\sum_{p=1}^{\infty}\left(\frac{\lambda^{p-1}\tau^{p\alpha-1}}{\Gamma(p\alpha)}\right)d\tau\right] \tag{63}$$

By reversing the order of the integral and the second summation and by factoring constants out of the integral this becomes

$$y_T(kT) = \sum_{j=1}^{k}\left[u((k-j)T)\sum_{p=1}^{\infty}\left(\frac{\lambda^{p-1}}{\Gamma(p\alpha)}\int_{(j-1)T}^{jT}\tau^{p\alpha-1}d\tau\right)\right] \tag{64}$$

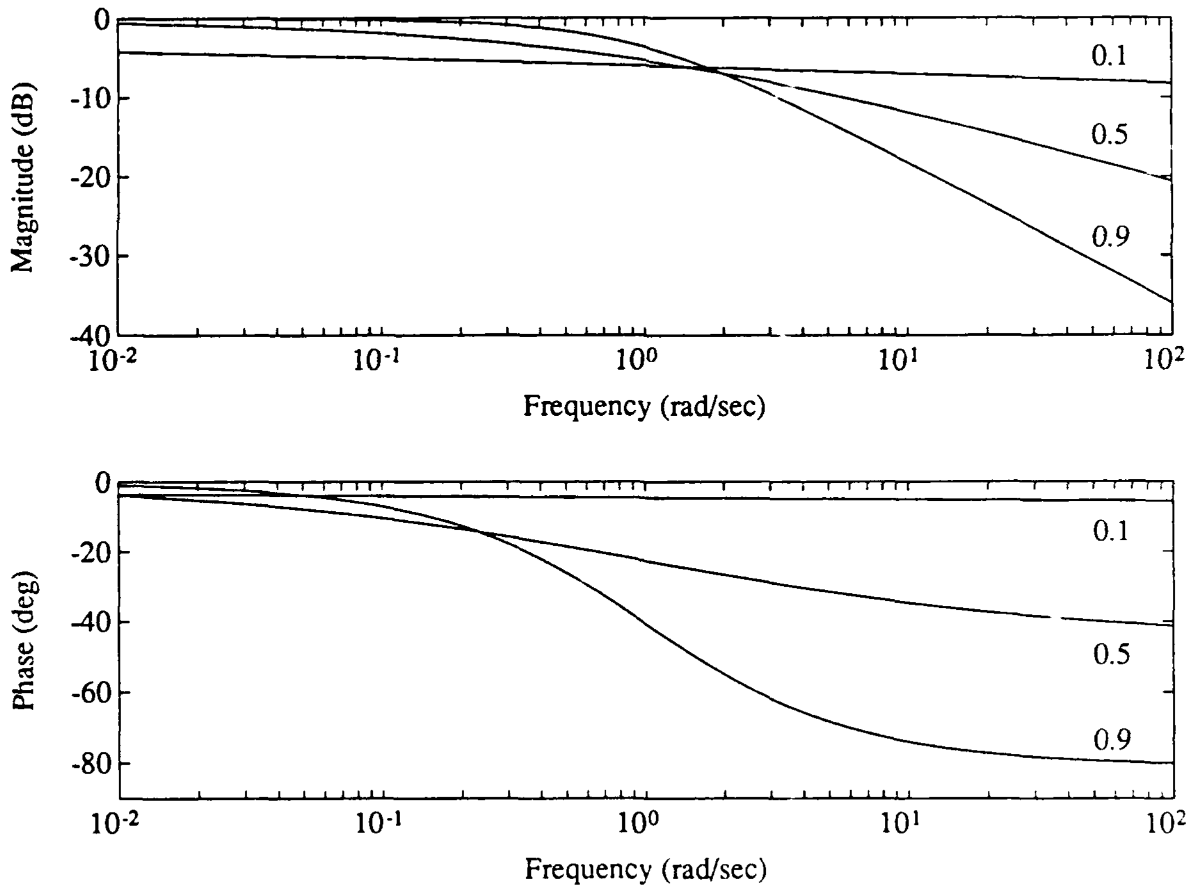


Figure 10 FDE Exact Frequency Responses

Performing the integration yields

$$y_T(kT) = \sum_{j=1}^{k}\left[u((k-j)T)\sum_{p=1}^{\infty}\left(\frac{\lambda^{p-1}}{\Gamma(p\alpha+1)}((jT)^{p\alpha} - ((j-1)T)^{p\alpha})\right)\right] \tag{65}$$

The second summation is now distributed over the two terms resulting from the integration and λ^{-1} is factored out to obtain

$$y_T(kT) = \sum_{j=1}^{k}\left[u((k-j)T)\frac{1}{\lambda}\left(\sum_{p=1}^{\infty}\frac{\lambda^p(jT)^{p\alpha}}{\Gamma(p\alpha+1)} - \sum_{p=1}^{\infty}\frac{\lambda^p((j-1)T)^{p\alpha}}{\Gamma(p\alpha+1)}\right)\right] \tag{66}$$

The last summations are recognized as forms of the Mittag-Leffler function so the convolution summation representation of the time response is

$$y_T(kT) = \sum_{j=1}^{k}\left\{u((k-j)T)\left[\frac{1}{\lambda}E_\alpha(\lambda^p(jT)^{p\alpha}) - \frac{1}{\lambda}E_\alpha(\lambda^p((j-1)T)^{p\alpha})\right]\right\} \tag{67}$$

and the pulse response sequence is

$$h_T(jT) = \left\{\frac{1}{\lambda}E_\alpha[\lambda(jT)^\alpha] - \frac{1}{\lambda}E_\alpha[\lambda((j-1)T)^\alpha]\right\} \tag{68}$$

Once again, the definition of the pulse response sequence points back to the representation of the unit pulse response as the sum of two unit step responses of opposite magnitude and separated by one sample period. Hence, this definition for the pulse response sequence could also be derived by solving the Mittag-Leffler representation for the time response, Eq (61), for a unit step input.

VI.5 Alternate Method

Using the pulse response sequence given above, the coefficients of the recursive difference equation can be found yielding a discrete representation of the original modal fractional differential equation. A drawback to this method is the infinite summation of the Mittag-Leffler function. If the coefficients of the discrete system are to be precalculated and will not change for a given application, the computer time required to approximate these functions with a large number of terms may not be problem. However, in parameter estimation and adaptive systems, new coefficients are often calculated in real-time or near real-time. The large number of terms required to approximate the Mittag-Leffler functions in those cases may overburden the computer.

An alternate method of calculating the coefficients can be used which significantly reduces the number of required calculations. The first step is to operate on both sides of the original modal equation with the alpha order fractional integral:

$$\mathrm{I}^{\alpha}\left[\mathrm{D}^{\alpha}y(t) - \lambda y(t)\right] = \mathrm{I}^{\alpha}[u(t)] \tag{69}$$

Using the law of exponents (discussed in Section II.2) and rearranging yields

$$y(t) = \mathrm{I}^{\alpha}[u(t) + \lambda y(t)] \tag{70}$$

The effect is to transform the system into a simple fractional integrator system, using the response output, y(t), as an input to the system. This can be visualized as proportional feedback on a simple block diagram.

The coefficients of the fractional integrator can be calculated based on the order and the sample time. Assuming the integrator input is u(t)+λy(t) from above, the standard difference equation of Eq (41) becomes

$$\begin{aligned} y_T(kT) = &- a_{n-1}y_T((k-1)T) - a_{n-2}y_T((k-2)T) - \cdots - a_0 y_T((k-n)T) \\ &+ b_{n-1}[u((k-1)T) + \lambda y_T((k-1)T)] \\ &+ b_{n-2}[u((k-2)T) + \lambda y_T((k-2)T)] \\ &\vdots \\ &+ b_0[u((k-n)T) + \lambda y_T((k-n)T)] \end{aligned} \tag{71}$$

Combining terms yields the standard difference equation format with new coefficients. The new coefficients, called $\hat{a}_i$ here to distinguish them from the simple integrator coefficients, are calculated using the simple integrator coefficients based on the formula

$$\hat{a}_i = a_i - \lambda b_i \tag{72}$$

The b_i coefficients calculated for the simple integrator do not change.

Using this method, a difference equation was derived for a half order modal differential equation with λ=-1. The sample time was 0.1 seconds (T=0.1) and 4 input and 4 output past values were used (n=4). The discrete transfer function for this system has the z-plane poles and zeros shown in Figure 11.

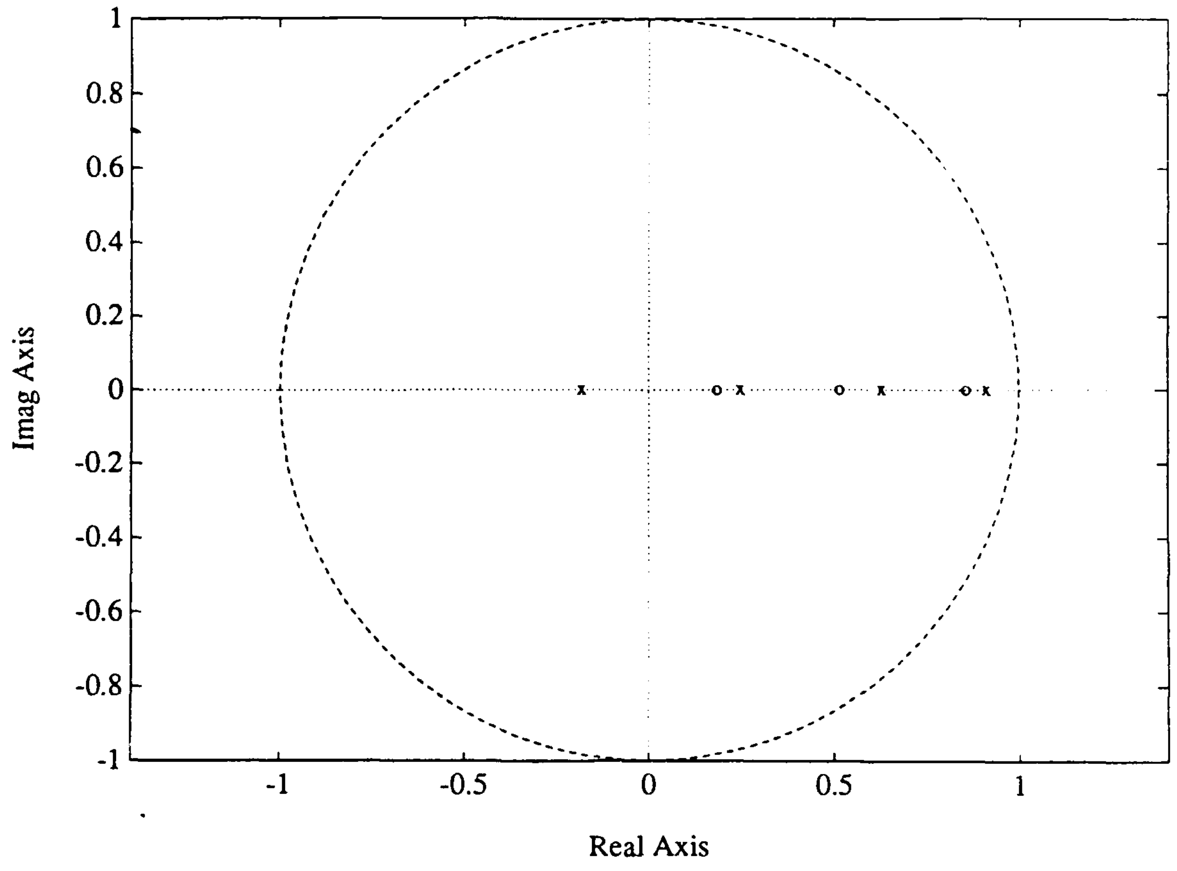


Figure 11 FDE Example z-Plane Pole-Zero Structure

The alternating poles and zeros on the real axis resemble the integrator example from Chapter 4.

Figure 12 shows the poles and zeros transformed into the continuous Laplace domain through the Tustin transformation. The sampling zero is again evident on the positive real axis at s=2/T. The other poles and zeros alternate on the negative real axis, but more than the other examples they are closer together, which tends to cancel out the effect of both on the frequency response.

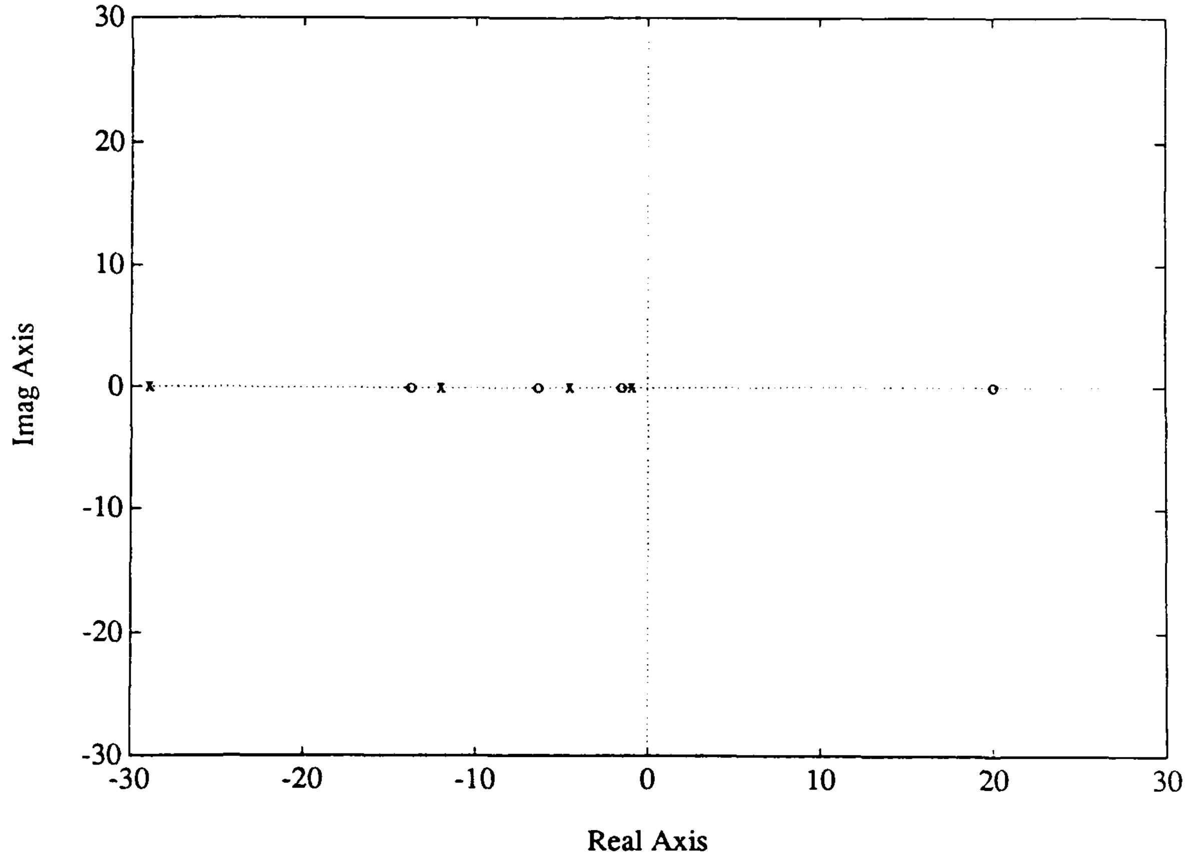


Figure 12 FDE Example s-Plane Pole-Zero Structure

In this case, the pole and zero locations themselves are not good indicators of the approximate passband. The pole near s=-29 would seem to indicate an upper limit near the Nyquist frequency of 31.4 rad/sec. As will be seen however, the upper limit is around 13 rad/sec using the criteria defined earlier. This value agrees with the earlier examples using a sampling period of 0.1 seconds. The poles nearer the origin are also poor indicators of passband limits since the frequency response shows a low frequency limit does not exist.

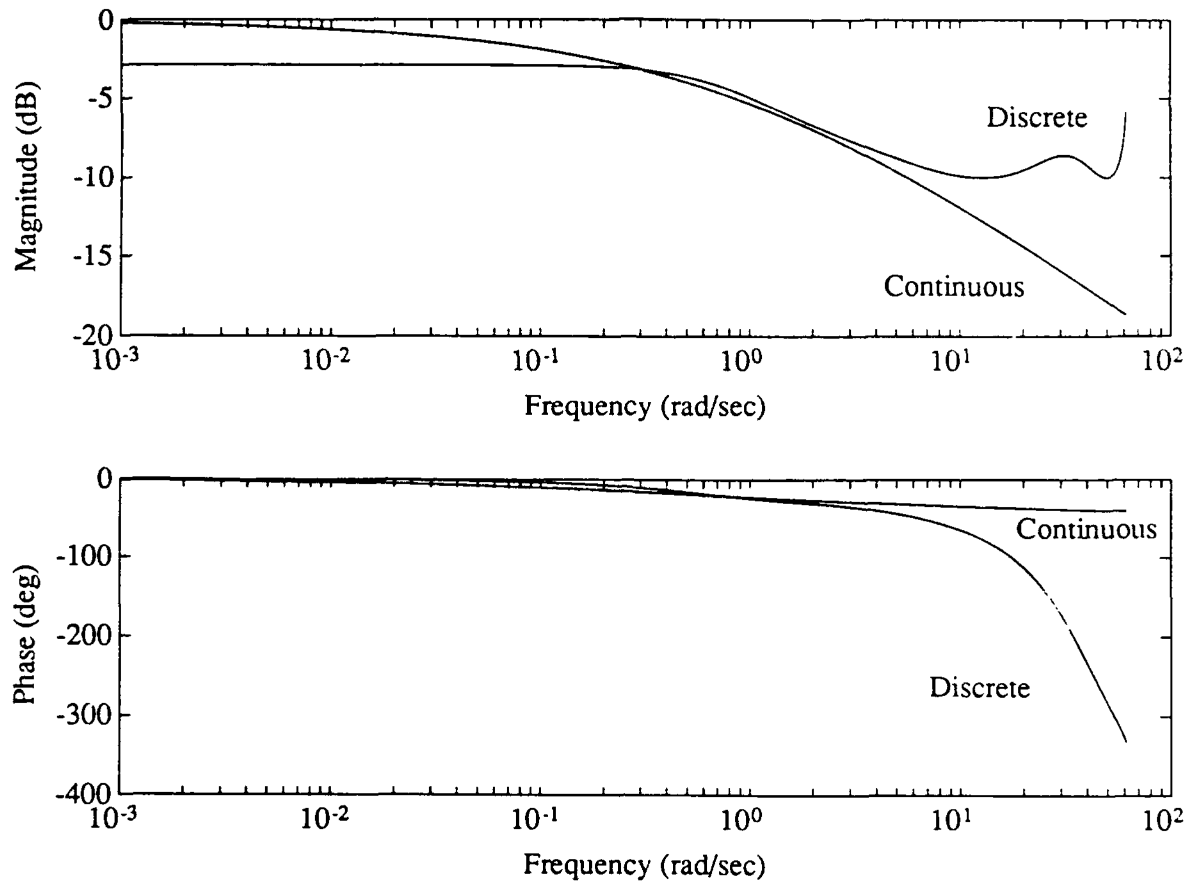


Figure 13 FDE Example Frequency Response

Figure 13 gives the frequency response of this discrete algorithm and the exact response of the continuous system. The low frequency continuous response is easily simulated since it tends toward zero magnitude or phase change; hence unlimited low frequency bandwidth can be achieved by simply restricting steady state error to less than 3 dB. (Phase error does not need to be specified; it is meaningless with respect to the step input implied by a steady state error specification.) The upper passband limit is once again influenced by the Nyquist criterion. One interesting note here is that 'he magnitude response exceeds the passband criteria at a lower frequency than the phase response, which is a change from previous examples.

Appendix B shows several time response plots for this discrete algorithm simulating the response of the example continuous FDE. The frequencies again are $\pi/4$, $\pi/2$, and π rad/sec. The time response plots confirm the accuracy of the frequency response plot at several points and show that the method is valid for this example.

The passbands for several values of sample time and number of retained terms are compared in Figure 14 for the case of $\alpha=0.5$ and $\lambda=-1$. As mentioned, the upper passband limits for any given sample time are on the same order as the integrator examples. The arrows indicate that no lower passband limit exists. The only cases which do not have unlimited low frequency bandwidth are those for $T=0.01$ using less than 20 retained past input and 20 retained past output values. The existence of unlimited low frequency performance for this case (using the defined criteria) using only a small number of terms offers wide frequency range flexibility of the algorithm, dependent mostly on sample time.

In this chapter, the modal fractional differential equation has been presented with its solution. Two methods of calculating the difference equation coefficients for a discrete representation have been derived, and an example problem was calculated and compared to the exact continuous frequency response. Successful discrete simulation of modal equations shows that any

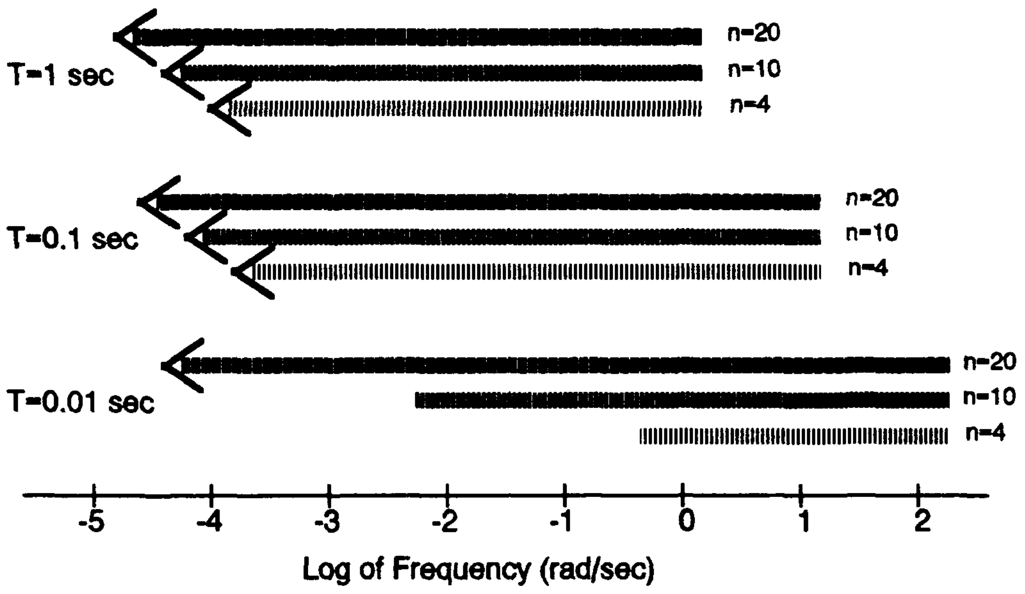


Figure 14 FDE Example Passbands

large system which can be decomposed into modal equations similar to Eq (52) is a candidate for discrete simulation using these methods.

VII. Conclusions and Recommendations

VI. 1 Conclusions

This investigation has successfully derived a method for discretely simulating the response of continuous fractional order systems. The pulse response sequence method of calculating difference equation coefficients is well-suited for extension to fractional order systems. The difference equations themselves are recursive in nature, requiring few calculations in each time step and little computer storage. These difference equations are easily implemented on a digital computer, for system simulation or real-time estimation and control.

Using passband criteria (± 3 dB in magnitude and ± 60 degrees in phase were used here) various cases can be compared to establish trends. It is clear that the Nyquist frequency (based on the sample time) determines the order of the high frequency passband limit in all cases, as expected given this sampled input system. For the simple fractional integrator, the sample time also establishes the low frequency passband limit for a given number of retained past values, allowing placement of the bandwidth where desired on the frequency spectrum.

The bandwidth for the simple integrator can be increased by increasing the number of retained past values. It is probably possible, theoretically, to simulate these cases down to whatever frequency is desired by approaching an infinite number of retained values, n. However, the cases plotted here indicate diminishing returns for increasingly larger values of n. Also, the Hankel matrices used in the calculation of the coefficients are ill-conditioned in most of the n=20 cases (with respect to the computer precision of 10^{-17} being used). The matrix condition number would be expected to get worse as n increased, so the practical limit on bandwidth, as defined, for the half order case seems to be about 3.5 decades for the simple integrator. Assuming n=20 is required to achieve this performance and the coefficients are precalculated, 40 multiplication operations and 39 addition operations are required each time step, and storage of 40 values is needed in memory.

The fractional order differential equation results obtained for the half order case and λ=-1 offer a much wider passband range since most cases had no lower limit, regardless of the number of retained terms. It would seem possible, then, to simply set the sample time based on the highest expected input frequency and be assured of lower frequency performance. However, the T=0.01 sec case varies from that pattern, not reaching unlimited low frequency performance until n=20, so that general rule does not hold in every case. For the cases not exhibiting a lower passband limit, there are interesting possibilities for large bandwidths using a small number of retained terms.

VI.2 Recommendations

One drawback of the method described here is that no analytic relationship has been produced relating the number of retained terms to the resulting bandwidth. Deriving such an expression would free a potential user from engaging in an iterative technique to minimize the number of retained values while meeting frequency response specifications. A possible starting point to find such an expression is to derive the exact z-transform for a fractional differintegral. A simple rule of thumb would also suffice, but would yield less insight into the problem.

Another drawback of this study is that it ignores initial conditions. If the algorithm is "turned on" at some time other that t=0 with an initial condition, this method will not given accurate results until the transients caused by that initial condition have sufficiently died out. To solve this problem, a discrete algorithm simulating the response of the system to initial conditions must be derived and the result added to the current algorithm.

If high accuracy response is desired for a certain frequency range, it may be worthwhile to investigate "tuning" the difference equation. Small changes in the pole or zero locations of the transfer function, corresponding to changes in the difference equation coefficients, can change the frequency response. It may be possible to move these pole and zero locations slightly to more closely match the exact continuous response for a given frequency or range of frequencies. Along this same line, there may be better methods of calculating the difference equation coefficients once the pulse response sequence is found, depending on the specific application.

A method of increasing bandwidth for all the discrete algorithms could be derived which operates at several different sampling rates. Several different sampling rates (where sampling rate is 1/T in Hz) would correspond to several different (probably overlapping) passbands on the frequency spectrum. This would allow a larger bandwidth for the system than would be possible for a single rate system.

The last recommendation concerns implementation of these difference equations. An interesting experimental follow-up study would implement these equations in a real-time computer program. Using integer order states as input signals, these algorithms would be able to estimate "in between" fractional states. Assuming the time functions of the integer order states used as inputs are relatively simple, the response of the discrete algorithms can be compared to the exact analytical results and possibly also compared to the results of analog devices.

Appendix A - Computer Programs

All calculations and most plots for this project were accomplished using the program Pro-Matlab (version 3.5i, 18 Jul 91, (c) The Mathworks, Inc.). The following is a sample of several command files and function definitions ("m-files") created to implement the methods derived in this investigation. Previously existing functions in the Matlab main program or any of the Matlab toolboxes are not repeated here. Note: The "dbode1" function is identical to Matlab's "dbode" function except that its phase angle calculation does not include an the unwrap function. The fractional order of integration or differentiation is referred to as β in these command files and function definitions, as opposed to α in the main body of the thesis.

FICRESP - Used to generate Figure 5 β=-0.5
T=0.1
n=8
w=logspace(-2,2,401)

```
%
% Command (script) file "ficresp"
%
% Command file to calculate the following for a fractional
%    integrator using the convolution summation
%    representation:
%
%    1) pulse response sequence, h (2n x 1)
%    2) coefficients of the finite difference equation, a
%                     and b (n x 1)
%    3) Hankel matrix of appropriate size (2n x 2n)
%    4) discrete transfer function polynomials, znum
%                     (n x 1)
%         and zden, (n+1 x 1)
%    5) algorithm response magnitude and phase,
%         almagdb and alphase (pts x 1)
%    6) exact magnitude and phase, exmagdb and exphase
%                     (n x 1)
%         (this is plotted vs. 5) on the screen)
% Note: the number of frequency points used, pts, is an
%                     input variable
%
% Before running, ensure the following variables are
%              defined:
%
%    1) beta, the order of integration (must be neg.)
%    2) T, the sample time in sec
%    3) n, the number of past input and output values to
%                     retain (2n total)
%    4) w, the frequency range of interest (pts x 1)
%
clear h a b H znum zden almag almagdb alphase
clear exmagdb exphase magerr phaseerr
%
% Calculate the pulse response sequence using 'prsgen'
%              function
%
h=prsgen(beta, 2*n, T);
%
% Calculate the discrete transfer function polynomials
%   using the pulse response sequence
%
zden(1)=1;
for k=1:n
 znum(k)=h(k);
 zden(k+1)=0;
end
%
% Calculate the algorithm frequency response magnitude
%   and phase using 'dbode1' function and convert
%     magnitude to dB
%
[almag,alphase]=dbode1(znum,zden,T,w);
almagdb=20*log10(almag);
%
% Calculate the exact frequency response using the
%              'derexact' function and calculate frequency response
%              errors
%
[exmagdb,exphase]=derexact(beta,w);
respgrf
```

FIZPLN - Used to generale Figure 6 β=-0.5
T=0.1
n=4

```
%
% Command (script) file "fizpln"
%
% Command file to calculate the following for a fractional
%        integrator:
%
%   1) pulse response sequence, h (2n x 1)
%   2) coefficients of the finite difference equation, a
%              and b (n x 1)
%   3) Hankel matrix of appropriate size (2n x 2n)
%   4) discrete transfer function polynomials, znum
%              (n x 1)
%   5) discrete poles and zeros, zpoles (n x 1), zzeros
%              (n-1 x 1)
%         (graphed on the screen)
%
% Before running, ensure the following variables are
%        defined:
%
%   1) beta, the order of integration (must be negative)
%   2) T, the sample time in sec
%   3) n, the number of past input and output values to
%              retain (2n total)
%
clear h a b H znum zden zzeros zpoles zgain
%
% Calculate the pulse response sequence using 'prsgen'
%        function
%
h=prsgen(beta, 2*n, T);
%
% Calculate the discrete transfer function coefficients
%     using 'coeff' function
%
[a,b,H]=coeff(h,n);
%
%  Calculate the discrete transfer function polynomials
%     using the 'disctf' function
%
[znum,zden]=disctf(a,b);
%
%  Calculate the discrete transfer function zeros, poles
%     and gain using 'tf2zp' function
%
[zzeros,zpoles,zgain]=tf2zp(znum,zden)
%
%  Draw a unit circle and map the poles and zeros
%               using 'pzmap'
%

axis([-1.2 1.2 -1 1])
t=0:.1:6.3;
plot(sin(t),cos(t),'--')
hold on
pzmap(zpoles,zzeros)
hold off
```

FDESPLN - Used to generate Figure 12 β=-0.5
T=0.1
n=4
w=logspace(-2,2,401)

```
%
% Command (script) file "fdespln"
%
% Command file to calculate the following for a fractional
%           differential equation:
%
%    1) pulse response sequence, h (2n x 1)
%    2) coefficients of the finite difference equation, a
%                 and b (n x 1)
%    3) Hankel matrix of appropriate size (2n x 2n)
%    4) discrete transfer function polynomials, znum
%                 (n x 1)
%    5) discrete poles and zeros, zpoles (n x 1), zzeros
%                 (n-1 x 1)
%    6) continuous poles and zeros, spoles and szeros
%         (graphed on the screen)
%
% Before running, ensure the following variables are
%           defined:
%
%     1) beta, the order of integration (must be negative)
%     2) T, the sample time in sec
%     3) n, the number of past input and output values to
%                 retain (2n total)
%     4) lambda, the modal equation eigenvalue
%
clear h a b H znum zden zzeros zpoles zgain
clear snum sden szeros spoles sgain
%
% Calculate the pulse response sequence using 'prsgen'
%           function
%
```

```
h=prsgen(beta, 2*n, T);
%
% Calculate the discrete transfer function coefficients
%   using 'coeff' function
%
[a,b,H]=coeff(h,n);
%
% Calculate the revised coefficients taking into account the
%   the input function of (lambda*y(t) + g(t))
%
for k=1:n
 a(k)=a(k)-(lambda*b(k));
end
%
% Calculate the discrete transfer function polynomials
%   using the 'disctf' function and print the discrete
%   poles and zeros
%
[znum,zden]=disctf(a,b);
[zzeros,zpoles,zgain]=tf2zp(znum,zden)
%
% Calculate the continuous transfer function polynomials
%   using 'd2cm' (Tustin transform method) and print the
%   continuous poles and zeros
%
[snum,sden]=d2cm(znum,zden,T,'tustin');
[szeros,spoles,sgain]=tf2zp(snum,sden)
%
% Print the continuous poles and zeros and plot them in
%   the s-plane using 'pzmap' and 'sgrid'
%
pzmap(snum,sden)
```

PRSGEN

```
function h=prsgen(beta,num,samptime)
%
% FUNCTION "prsgen"  - generates pulse response sequence
%                     using piecewise-constant method
%
%       h=prsgen(beta,num,samptime)
%
for k=1:num
  h(k)=((k*samptime)^(-beta)-((k-1)*samptime)^(-beta))
            /gamma(1-beta);
end
```

DEREXACT

```
function [mag,phase]=derexact(beta,w)
%
% FUNCTION "derexact"  - generates the exact phase and
%       magnitude response for a simple differintegrator
%      given beta and the input freq's (r/s)
%
%       [mag,phase]=derexact(beta,w)
%
num=length(w);
for k=1:num
  g(k)=(w(k)*i)^beta;
  mag(k)=20*log10(abs(g(k)));
  phase(k)=(180/pi)*atan2(imag(g(k)),real(g(k)));
end
```

RESPGRF

```
%
% Command (script) file "respgrf"
%
% Command file to plot the algorithm magnitude
%   and phase response vs. the exact magnitude
%      and phase response
%
% The following variables must be defined:
%    1) w, frequencies of interest (rad/sec)
%    2) almagdb, algorithm magnitude response over w (dB)
%    3) alphase, algorithm phase response over w (deg)
%    4) exmagdb, exact magnitude response over w (dB)
%    5) exphase, exact phase response over w (deg)
%
subplot(211), semilogx(w,exmagdb)
axis;
subplot(211), semilogx(w,almagdb)
xlabel('Frequency (rad/sec)')
ylabel('Magnitude (dB)')
axis;
subplot(212), semilogx(w,alphase)
axis;
subplot(212), semilogx(w,exphase)
xlabel('Frequency (rad/sec)')
ylabel('Phase (deg)')
axis;
subplot(111)
```

COEFF

```
function [acoeff,bcoeff,H]=coeff(h,num)
%
% FUNCTION "coeff"  - finds finite difference equation coefficients
%                     and also returns the Hankel matrix
%
%    [acoeff,bcoeff,H]=coeff(h,num)
%
clear H acoeff bcoeff
%
%  Calculate Hankel matrix and solve for a coefficients
%
H=hankel(h(1:num),h(num:2*num-1));
acoeff=H\[-h(num+1:2*num)]';
acoeff=acoeff';
%
%  Calculate b coefficients
%
bcoeff(num)=h(1);
for k=1:num-1
 sum=0;
 for j=1:k
  sum=sum+(h(j)*acoeff(num-k+j));
 end
 bcoeff(num-k)=sum+h(k+1);
end
bcoeff(1:num)=[bcoeff(1:num)]';
```

DISCTF

```
function [numpoly,denpoly]=disctf(acoeff,bcoeff)
%
% FUNCTION "disctf" - finds the discrete transfer
%      function numerator and denominator polynomials
%      in decreasing powers of z
%
%     [numpoly,denpoly]=disctf(acoeff,bcoeff)
%
num=length(acoeff);
numpoly=[bcoeff(num:-1:1)];
denpoly=[1,acoeff(num:-1:1)];
```

FDEEXACT

```
function [mag,phase]=fdeexact(beta,lambda,w)
%
% FUNCTION "fdeexact" - generates the exact phase and
%              magnitude response for a fractional differential
%              equation given beta, lambda, and the input
%              freq's (r/s)
%
%     [mag,phase]=fdeexact(beta,lambda,w)
%
num=length(w);
for k=1:num
  g(k)=1/((w(k)*i)^beta-lambda);
  mag(k)=20*log10(abs(g(k)));
  phase(k)=(180/pi)*unwrap(atan2(imag(g(k)),real(g(k))));
end
```

FINDBW

```
%
% Command (script) file "findbw"
%
% Command file to calculate the bandwidth (based on
%           magnitude and phase) for a discrete fractional
%           differintegral algorithm
%
% Outputs:
%   1) magbw - the bandwidth based on magnitude
%                 error of 3 dB
%   2) phsbw - the bandwidth based on phase error of 60 deg
%
% Ensure the following are defined before execution:
%   1) magerr - the magnitude error vector in dB (pts x 1)
%   2) phserr - the phase error vector in deg (pts x 1)
%   3) w - the frequency vector in rad/sec (pts x 1)
%
clear magerri phserri num diffm diffp indexm diffp
clear maxmagi maxphsi magbw phsbw
magerri=find(magerr<3);
phserri=find(phserr<60);
numm=length(magerri);
for j=2:numm
 diffm(j-1)=magerri(j)-magerri(j-1);
end
indexm=find(diffm>1);
maxmagi=magerri(min(indexm));
if isempty(maxmagi)
 maxmagi=max(magerri);
end
magbw=log10(w(maxmagi))-log10(w(magerri(1)));
nump=length(phserri);
for j=2:nump
 diffp(j-1)=phserri(j)-phserri(j-1);
```

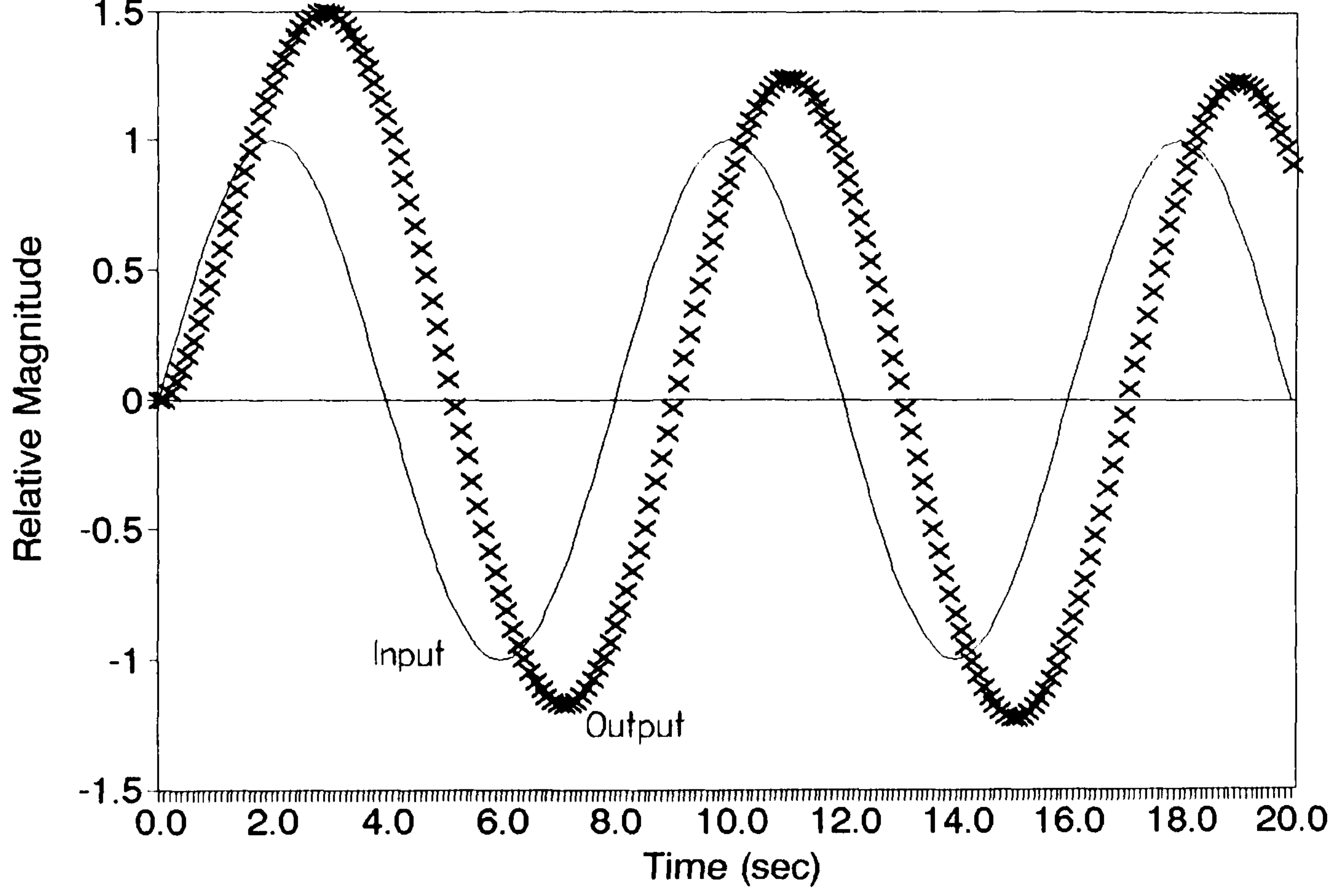
```
end
indexp=find(diffp>1);
maxphsi=phserri(min(indexp));
if isempty(maxphsi)
  maxphsi=max(phserri);
end
phsbw=log10(w(maxphsi))-log10(w(phserri(1)));
if w(magerri(1))==w(1)
  disp('Lower end of mag bw not found.')
end
if w(maxmagi)==max(w)
  disp('Upper end of mag bw not found.')
end
if w(phserri(1))==w(1)
  disp('Lower end of phs bw not found.')
end
if w(maxphsi)==max(w)
  disp('Upper end of phs bw not found.')
end
magbw=[w(magerri(1)),log10(w(magerri(1)));w(maxmagi),log10(w(maxmagi))]
phsbw=[w(phserri(1)),log10(w(phserri(1)));w(maxphsi),log10(w(maxphsi))]
```

Appendix B - Time Response Plots

These time response plots were generated by propogating the recursive difference equation forward through 200 time steps, corresponding to a 20 second simulation of the continuous fractional order system (since T=0.1 sec in all examples). The difference equations were implemented and graphed using the Quattro Pro spreadsheet program (version 1.0, (c) Borland, Inc.)

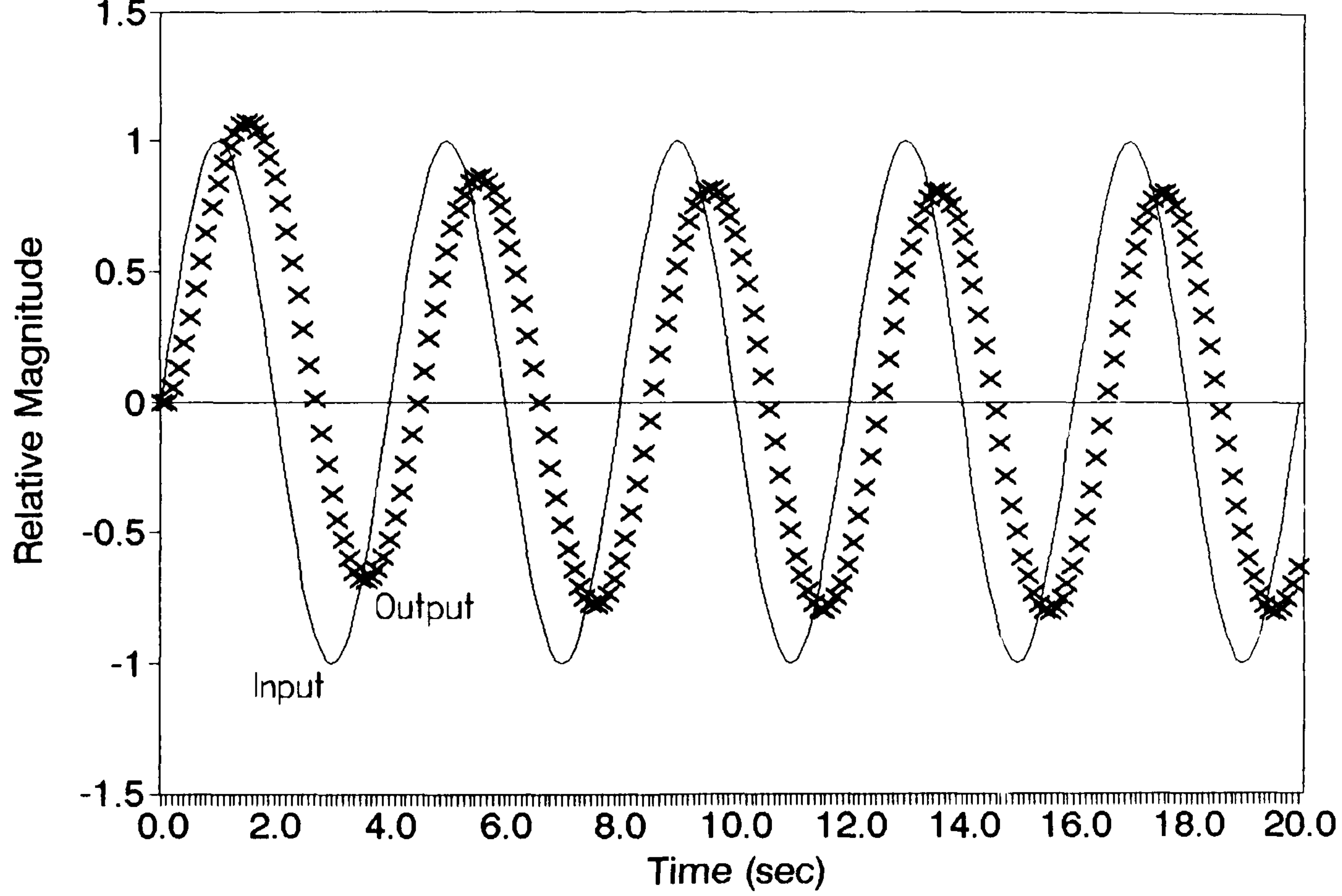
Recursive Difference Algorithm

Fractional Integration Example



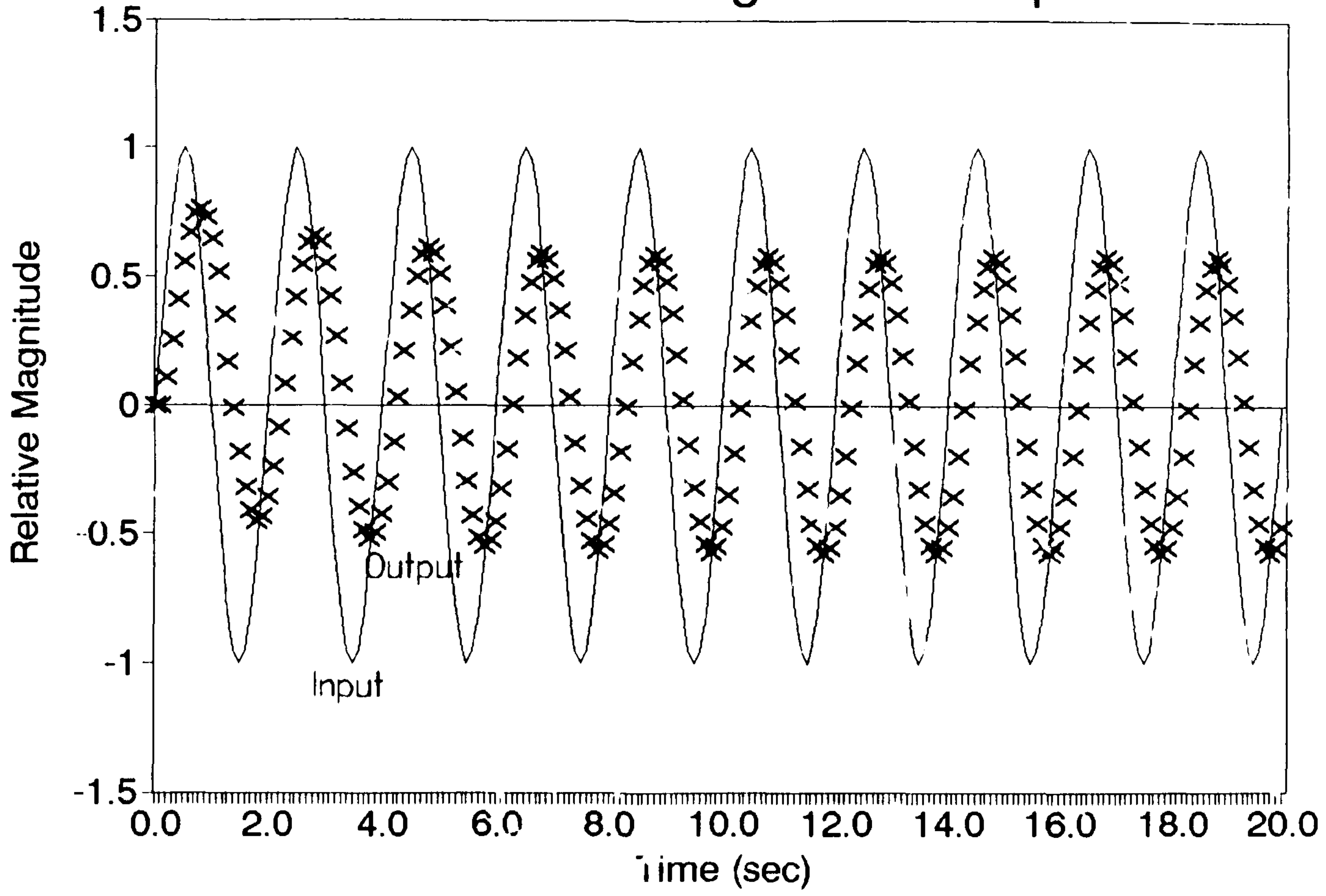
Recursive Difference Algorithm

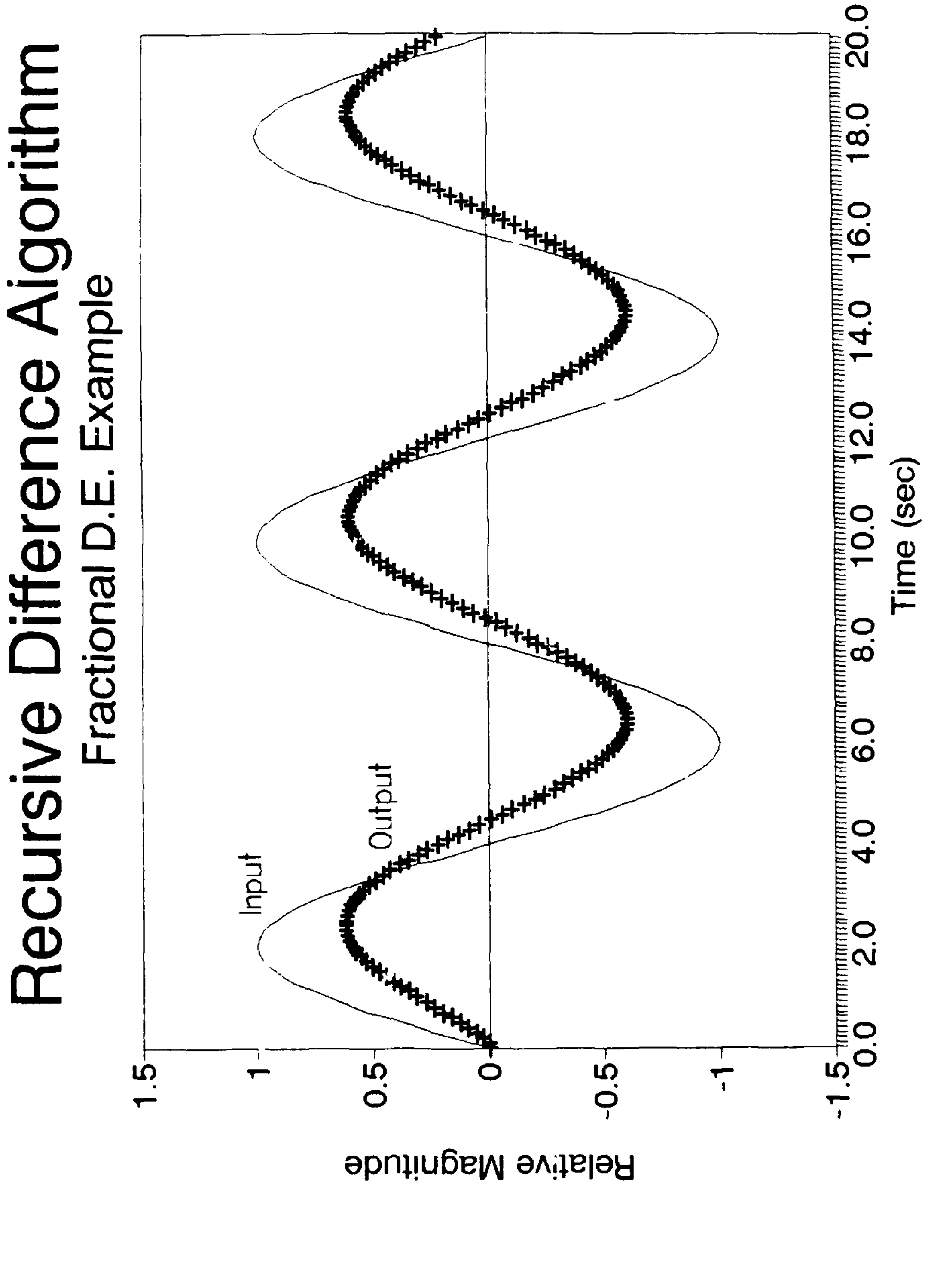
Fractional Integration Example



Recursive Difference Algorithm

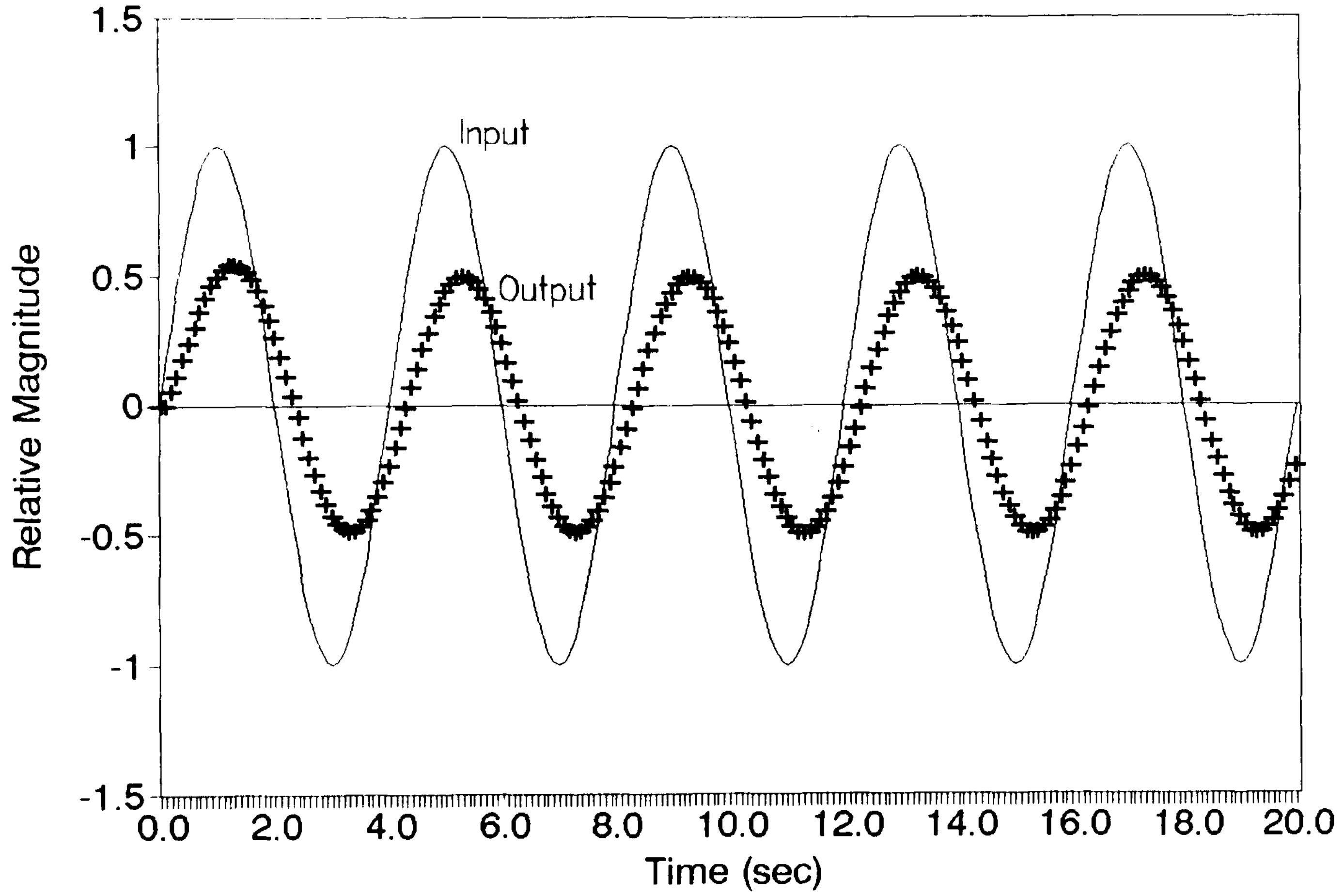
Fractional Integration Example




Recursive Difference Aigorithm
Fractional D.E. Example
Input
Output
Relative Magnitude
1.5
1
0.5
0
-0.5
-1
-1.5
0.0
2.0
4.0
6.0
8.0
10.0
12.0
14.0
16.0
18.0
20.0
Time (sec)

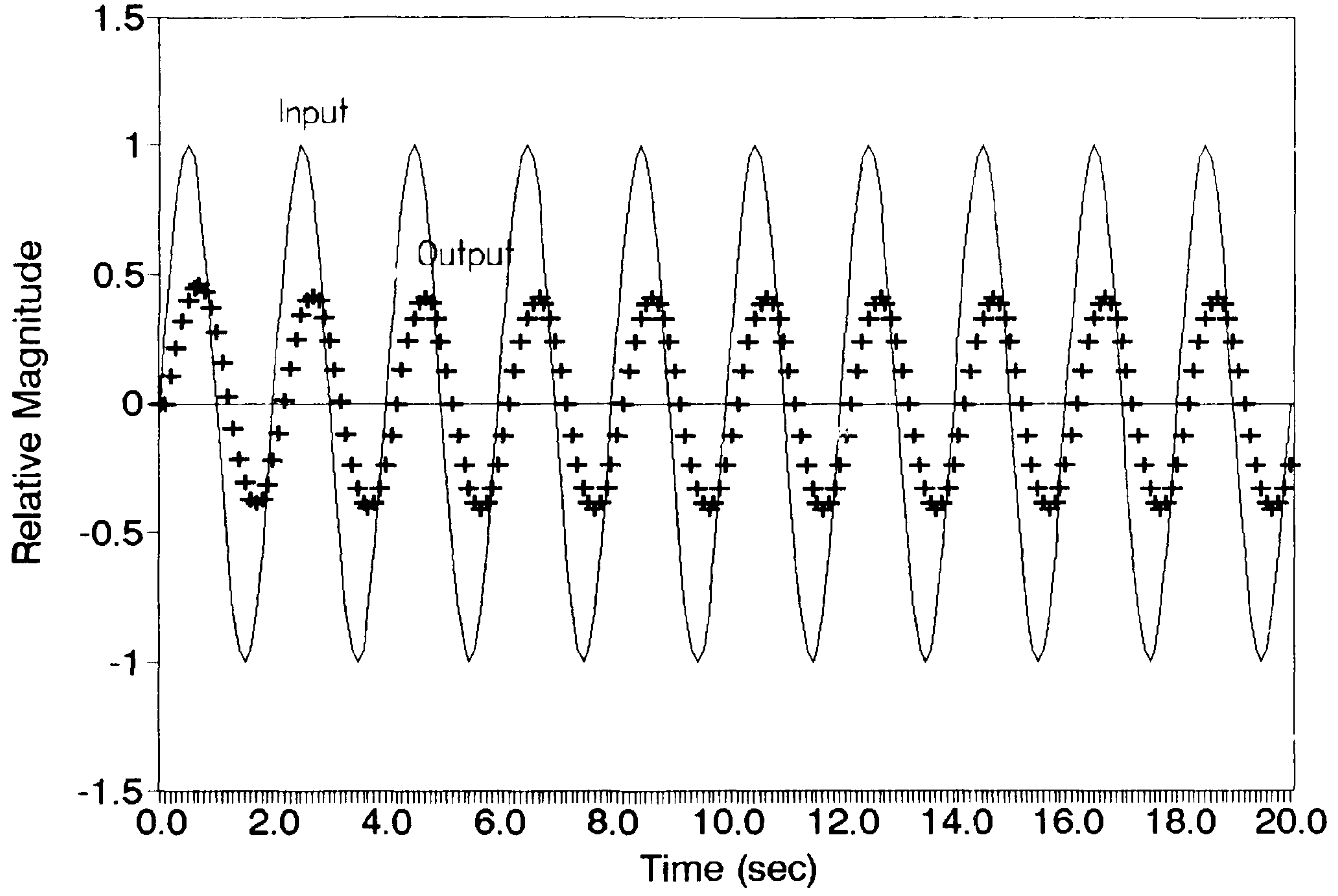
Recursive Difference Algorithm

Fractional D.E. Example



Recursive Difference Algorithm

Fractional D.E. Example



Bibliography

1. Bagley, Ronald L. "On the Fractional Order Initial Value Problem and Its Engineering Applications," *Fractional Calculus and Its Applications, International (Tokyo) Conference Proceedings.* 12-20. Tokyo: College of Engineering, Nihon University, 1990.

2. Bagley, Ronald L. *The Initial Value Problem for Fractional Order Differential Equations with Constant Coefficients.* AFIT-TR-EN-88-1 (2nd Edition). School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, September 1989.

3. Bagley, R.L. and R.A. Calico. "Fractional Order State Equations for the Control of Viscoelastically Damped Structures," *Journal of Guidance, Control, and Dynamics. 14*:304-311 (March-April 1991).

4. Bagley, Ronald L. and Peter J. Torvik. "A Generalized Derivative Model for an Elastomer Damper," *The Shock and Vibration Bulletin. 49*:135-143 (September 1979).

5. Franklin, Gene F. and J. David Powell. *Digital Control of Dynamic Systems.* Reading, MA: Addison-Wesley, 1981.

6. Houpis, Constantine H. and Gary B. Lamont. *Digital Control Systems - Theory, Hardware, Software.* New York: McGraw-Hill, 1985.

7. Klonoski, Capt Kevin D. *Fractional Order Feedback in Linear Systems.* MS Thesis, AFIT/GA/ENY/88D-05. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 1989.

8. National Bureau of Standards. *Handbook of Mathematical Functions.* Applied Mathematic Series 55. Washington: Government Printing Office, December 1972.

9. Ogata, Katsukiko. *Modern Control Engineering.* Englewood Cliffs, NJ: Prentice-Hall, 1970.

10. Oldham, Keith B. and Jerome Spanier. *The Fractional Calculus.* Orlando: Academic Press, 1974.

11. Reid, J. Gary. *Linear System Fundamentals.* New York: McGraw-Hill, 1983.

12. Ross, Bertram. "Fractional Calculus," *Mathematics Magazine, 50.* 115-122 (May 1977).

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 1991	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE
Discrete Simulation Of Fractional Order Systems

5. FUNDING NUMBERS

6. AUTHOR(S)
Jeffrey A. Blank, Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GA/ENY/91D-15

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Mr. Jerome Pearson
WL/FIBG
WPAFB OH 45433

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Fractional calculus has been shown useful for describing many real world systems, and studies are currently underway to generalize control theory to incorporate fractional states. This investigation derives a method for simulating the time response of fractional order systems using a recursive difference equation. The technique used effectively approximates a simple fractional order integrator as a summation of integer order terms. The discrete transfer function is also derived and the frequency response of the discrete algorithm is compared to the exact continuous case. Using 20 or more retained past values in the difference equation, the discrete half-order integrator demonstrates a passband of more than three decades. A slightly modified method is used to derive a recursive difference equation which simulates the response of a modal fractional order differential equation. Frequency response analysis of an example having an eigenvalue of -1 shows the characteristics of a low pass filter, effectively simulating the continuous system response for all frequencies below the Nyquist limit, even when using a small number of retained past values.

14. SUBJECT TERMS
Fractional Calculus, Fractional Order Differential Equations, Discrete Simulation, Discrete Modelling, Digital Control, Sampled-Data Systems, State Estimation

15. NUMBER OF PAGES
88

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL