A NEW FINITE DIFFERENCE SOLUTION TO THE FOKKER-PLANCK EQUATION WITH APPLICATIONS TO PHASE-LOCKED LOOPS

A THESIS

Presented to

The Faculty of the Division of Graduate

Studies and Research

By

William Mulherin O'Dowd, Jr.

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

in the School of Electrical Engineering

Georgia Institute of Technology

November 1973

A NEW FINITE DIFFERENCE SOLUTION TO THE FOKKER-PLANCK

EQUATION WITH APPLICATIONS TO PHASE-LOCKED LOOPS

Approved:

[illegible]

J. L. Hammond, Jr., Chairman

[illegible]

A. S. Debs

[illegible]

G. H. Meyer

Date approved by Chairman: 1/4/74

ACKNOWLEDGMENTS

I would like to thank Dr. Joseph L. Hammond, Jr. for suggesting the thesis problem and for his guidance, encouragement, and help throughout the research. I would like to express my appreciation to Drs. A. S. Debs and G. H. Meyer for their services as members of the reading committee.

I would also like to thank Mrs. Lydia Geeslin and Mrs. Betty Yarborough for the excellent job they did in typing this thesis.

TABLE OF CONTENTS

TABLE OF CONTENTS (Concluded)

LIST OF TABLES

LIST OF TABLES (Concluded)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Continued)

LIST OF ILLUSTRATIONS (Continued)

LIST OF ILLUSTRATIONS (Concluded)

SUMMARY

A modified algorithm is developed which greatly improves the efficiency of solving parabolic partial differential equations. The main emphasis of the work is in solving a certain class of these problems, the Fokker-Planck equations for systems operating in the presence of noise. For mathematical convenience and to enhance the accuracy of the solutions, the Fokker-Planck equations are expressed in terms of probability distribution functions rather than probability density functions.

The modified algorithm combines an explicit finite difference scheme and polynomial interpolation in such a way as to greatly reduce the amount of information that has to be stored and the number of numerical operations required in order to obtain the solution to a Fokker-Planck equation. For one-dimensional equations, at any time the modified algorithm stores the data at every P^{th} point in the grid of the standard explicit scheme. In order to advance the solution forward in time, the additional information required by the explicit method is generated by fitting polynonials of order q to the stored data. Since data is stored for every P^{th} grid point, it is only necessary to compute the solutions at these points. It is shown that the modified algorithm is consistent, convergent, stable, and has the same order of accuracy as the original explicite scheme if $q \geq 2$.

The modified algorithm is tested and its parameters selected by solving the Fokker-Planck equations for several linear systems. This approach is taken so that the theoretical solutions, which can be obtained for linear systems, can be compared to the numerical results. It is shown that second order polynomial interpolation ($q = 2$) gives the best results. A selection of $P = 5$ (store 1/5 of the data in each space dimension) is made by observing the trade-offs between accuracy and the amounts of computer time and storage required for different values of P.

With the parameters selected ($q = 2$, $P = 5$), the modified algorithm gives accurate solutions and yields large savings in computer time and storage. The amount of savings in computer storage realized for one, two, and three-dimensional equations is 80%, 96%, and greater than 99% respectively. The corresponding savings in computer time for one- and two-dimensional problems is about 70% and about 80% respectively. The savings in computer time for three-dimensional equations is conservatively estimated to be in excess of 80%.

A three-dimension Fokker-Planck equation for a third order linear system is solved using the modified algorithm. This example clearly illustrates that the large saving in computer time and storage obtained with the modified algorithm makes it possible for this method to solve problems that would otherwise be impractical to solve.

The modified algorithm is used to obtain complete solutions to the Fokker-Planck equations for a first and second order phase-locked loop. The solutions, which are sought on modulo 2π, are started from initial conditions which are uniformly distributed and are run to steady state. Results are obtained for several different signal to noise ratios. The steady state results for the first order loop agree with the theoretical solutions, and the steady state results for the second order loop agree with a set of experimental solutions which appear in the literature.

The modified algorithm is also used to solve Fokker-Planck equations for a first and second order gated phase-locked loop. This problem arises in a Time Division Multiple Access (TDMA) system which uses Phase Shift Keyed (PSK) modulation and which requires that phase coherence be maintained from burst to burst. The objective of these simulations is to find the steady state variances of the phase errors of the systems. It is observed that the steady state variances of the phase errors for both the first and second order gated loops can be found (estimated for the second order loop) without obtaining complete solutions to the Fokker-Planck equations. This is a helpful result since solutions to these problems for systems with practical duty factors require large amounts of computer time.

CHAPTER I

INTRODUCTION

The Fokker-Planck equation originally evolved from the study of Brownian motion. Brownian motion was first observed by Robert Brown in 1828, and was first correctly explained by Albert Einstein in 1905. In the ensuing twenty-five years many scientists studying the area (Smoluchowski, Fokker, Planck, Ornstein, Burger, Firth, et al.) realized that the probability density function for the position of a particle undergoing Brownian could be described by a parabolic partial differential equation which became commonly known as the Fokker-Planck equation.

A major breakthrough in the Brownian motion problem was presented by Uhlenbeck and Ornstein [29] in 1930. They arrived at the Fokker-Planck equation by considering the equation of motion of the particle, which is known as the Langevin equation. This was not mathematically rigorous, and it was not until many years later that Doob [16], using Ito calculus, justified the work mathematically.

Another major accomplishment appeared in 1931 when Kolmogorov [43] presented Kolmogorov's forward and backward equations. The forward equation is more generally known as the Fokker-Planck equation. As their names imply, the forward and backward equations are adjoint equations, the forward equation having to be solved forward in time, and the backward equation in reverse time. The backward equation is of little use since a solution of it would require boundary conditions that include a known solution at an advanced time.

As it is used in this dissertation, the Fokker-Planck equation relates the statistical properties of the state variables of a system to the statistical properties of the inputs to the system and the characteristics of the system. In particular, the Fokker-Planck equation is an n-dimensional second order parabolic partial differential equation whose solution is the joint probability density function of the n states of the system. The development of the Fokker-Planck equation requires that states of the system be a continuous vector Markov process [39]. This requires that the system have a white noise input. Other considerations restrict the noise inputs to be Gaussian white noise [10].

Consider the system illustrated in Figure 1. The state equations of the system are

$$\dot{\underline{y}} = g(\underline{y},\underline{\eta},t) \ , \tag{1-1}$$

where the underscored variables are column vectors with n components. The input to the system, $\underline{\eta}(t)$, is a column vector whose components are all Gaussian white noises (not necessarily stationary).

Under mild conditions of continuity of the joint probability density function of the states of the system and its first few derivatives, the Fokker-Planck equation can be derived [10,19,6]. The Fokker-Planck equation for a general nth order system, (1-1), is

$$\frac{\partial f(y_1,. \ . \ .,y_n,t/x_1,. \ . \ .,x_n,t_0)}{\partial t} = \tag{1-2}$$

(continued)

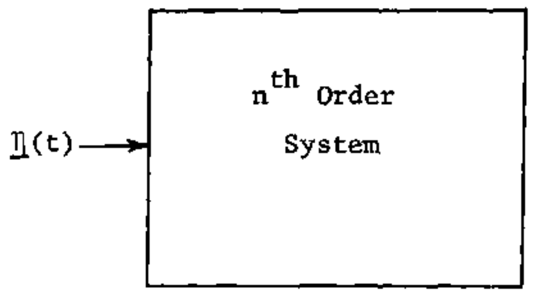


Figure 1. General n^{th} Order System

$$+ \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial^2 [C_{ij}(y_1,\ldots,y_n,t) f(y_1,\ldots,y_n,t/x_1,\ldots,x_n,t_0)]}{\partial y_i \partial y_j}$$

$$- \sum_{i=1}^{n} \frac{\partial [C_i(y_1,\ldots,y_n,t) f(y_1,\ldots,y_n,t/x_1,\ldots,x_n,t_0)]}{\partial y_i} ,$$

where $f(y_1,\ldots,y_n,t/x_1,\ldots,x_n,t_0)$ is the joint probability density function of the state of the system conditioned on the initial state of the system. The coefficients $C_{i,j}(y_1,\ldots,y_n,t)$ and $C_i(y_1,\ldots,y_n,t)$, commonly called the moments of the Fokker-Planck equation, are defined by

$$C_i(y_1,\ldots,y_n,t) = \lim_{\Delta t \to 0} \frac{E(\Delta y_i)}{\Delta t} , \tag{1-3}$$

and

$$C_{ij}(y_1,\ldots,y_n,t) = \lim_{\Delta t \to 0} \frac{E(\Delta y_i \Delta y_j)}{\Delta t} ,$$

where $E(\Delta y_i)$ is the expected value of the incremental change in y_i.

A rigorous development of the Fokker-Planck equation requires sophisticated theory of stochastic processes. In the current literature the development of the Fokker-Planck equation is approached in a variety of ways and with all different degrees of difficulty. Some of the more rigorous and difficult works are those presented by Doob [18], Feller [17], Dykin [4], Ito [21], and Gnedenko [20]. Some of the more easily understood developments are those of Stratonovich [19], Bharucha-Reid [10], Middleton [2], and Morgan [6].

In order to simplify notation, for the remainder of this work the probability density function in Fokker-Planck equations will not be written as being conditioned on the initial probability density of the states of the systems. This is reasonable since it is obvious that the initial probability density is required in order to solve the Fokker-Planck equation. Therefore, the conditioning on the initial state will be assumed and will not be explicitly written as such.

In order to illustrate the Fokker-Planck equation, consider the system shown in Figure 2. The state equation for this system is

$$\dot{x} = - \beta(x,t) + \eta(t) \; . \tag{1-4}$$

The input, $\eta(t)$, is Gaussian white noise with a power spectral height of N_0. The moments of the Fokker-Planck equation, defined by (1-3), are

$$C_1 = \lim_{\Delta t \to 0} \frac{E(\Delta x)}{\Delta t} = - \beta(x,t) \; , \tag{1-5}$$

and

$$C_{11} = \lim_{\Delta t \to 0} \frac{E(\Delta x^2)}{\Delta t} = \frac{N_0}{2} \; .$$

Therefore, the Fokker-Planck equation which describes the probability density function of the state of the system, $x(t)$, is

$$\frac{\partial f(x,t)}{\partial t} = \frac{\partial}{\partial x} (\beta(x,t) \; f(x,t)) + \frac{N_0}{4} \frac{\partial^2 f(x,t)}{\partial x^2} \; . \tag{1-6}$$

Phase-Locked Loops

A significant engineering application of the Fokker-Planck equation is in the analysis of phase-locked loops in the presence of noise [8,9, 23]. The phase-locked loop is a very practical device having found a

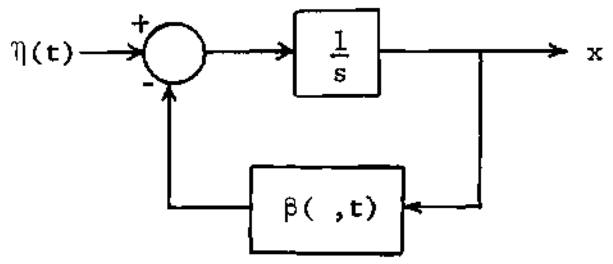


Figure 2. First Order System

variety of uses in recent years. They are used extensively in such applications as radar-tracking, missile guidance and navigation, and synchronization and detection in phase-coherent communication systems. Phase-locked loops are now to the point of being mass produced in modular form for many everyday uses in communication systems.

The purpose of the phase-locked loop is to track the phase of a received signal with a reference signal. The automatic phase control system commonly used is illustrated in Figure 3. The variables in and related to Figure 3 are defined as:

VCG	- voltage controlled generator,
$\sqrt{2}\, A \cos(\theta(t))$	- received signal,
$\sqrt{2}\, K_1 \sin(\theta'(t))$	- reference signal,
K_2	- frequency sensitivity constant of the VCG ($\omega_{VCG} = \omega_0 + K_2 e(t)$) ,
ω_0	- quiescent frequency of the VCG,
$h(t)$	- impulse response of the linear filter,
$\phi(t)$	- phase error ($\theta(t) - \theta'(t)$).

The system attempts to adjust $\theta'(t)$ until it is equal to $\theta(t)$. When this is accomplished, the signals are said to be phase-coherent or in phase-lock.

The dynamic response of the phase-locked loop is described by

$$\frac{d\phi(t)}{dt} = \frac{d\theta(t)}{dt} - \omega_0 - AK_1K_2 \int_0^t h(t-u) \sin(\phi(u))du \ . \qquad (1\text{-}7)$$

This equation is somewhat simplified by letting

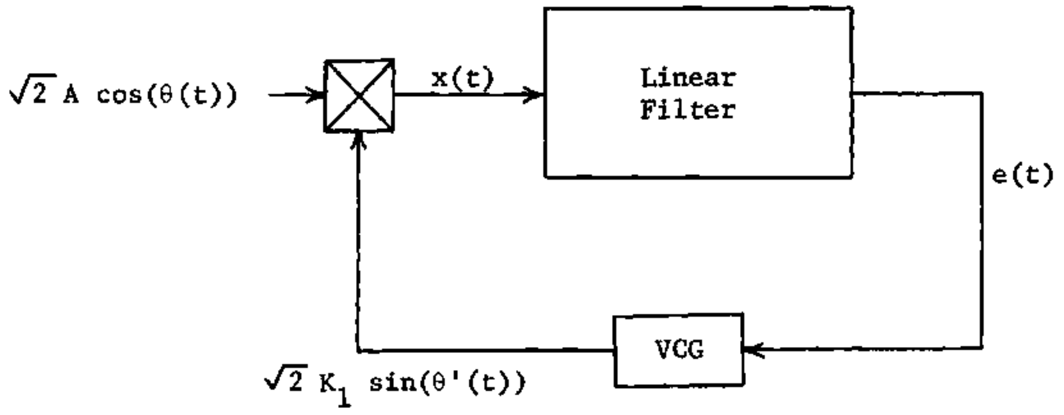


Figure 3. Phase-Locked Loop

$$K = K_1K_2 \ , \qquad (1\text{-}8)$$

$$\theta_1(t) = \theta(t) - \omega_0 t \ ,$$

and

$$\theta_2(t) = \theta'(t) - \omega_0 t \ .$$

When (1-8) is put into (1-7) , (1-7) becomes the equation commonly used to describe the phase-locked loop. It is

$$\frac{d\phi(t)}{dt} = \frac{d\theta_1(t)}{dt} - AK\int_0^t h(t-u)\ \sin(\phi(u))du \ . \qquad (1\text{-}9)$$

This equation suggests the general block diagram for the phase-locked loop (illustrated in Figure 4).

The order of a phase-locked loop is defined as the order of the differential equation describing the loop, (1-9). If the linear filter in the forward path is an n^{th} order filter, the system is an $n+1^{st}$ order phase-locked loop. The VCG adds the additional pole to the system. Therefore, a first order phase-locked loop has no linear filter present.

Phase-Locked Loop in the Presence of Noise

A very interesting and difficult problem which has attracted a lot of attention in recent years is the operation of phase-locked loops in the presence of noise. Most communication systems are disturbed by thermal noise, which is a zero-mean wideband Gaussian process which has a power spectral density that is nearly flat over the frequency range of the receiving equipment (Gaussian white noise for all practical purposes).

Let the received signal be

$$2\sqrt{A}\cos\ (\theta(t)) + \eta(t) \ , \qquad (1\text{-}10)$$

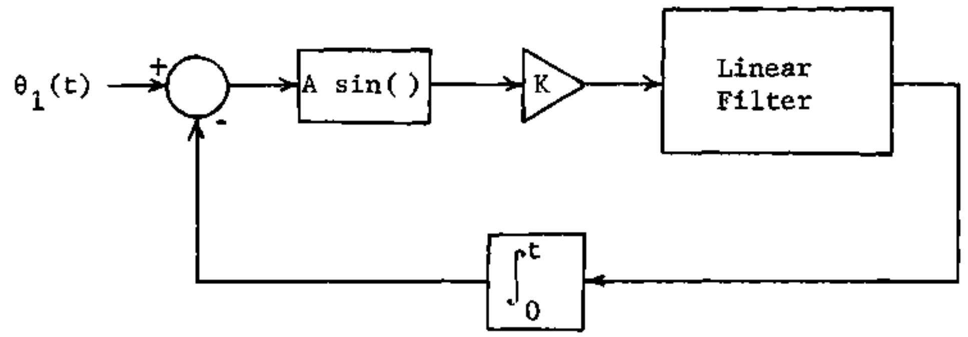


Figure 4. Block Diagram of Phase-Locked Loop

where $\eta(t)$ is Gaussian white noise with a power spectral height of N_0. The differential equation which now describes the dynamic response of the phase-locked loop is

$$\frac{d\phi(t)}{dt} = \frac{d\theta_1(t)}{dt} - K \int_0^t (A \sin(\phi(u)) + \eta'(u))h(t-u)du \ . \qquad (1\text{-}11)$$

In this representation it is assumed that, for mathematical convenience, $\eta(t)$ has been passed through a symmetric wideband bandpass filter with center frequency ω_0 (quiescent setting of VCG) and a flat passband which passes only frequencies below $2\omega_0$. The result of this filtering is $\eta'(t)$. Due to the low pass filtering present in the phase-locked loop, $\eta'(t)$ still looks like Gaussian white noise to the system. Figure 5 illustrates the block diagram for the phase-locked loop in the presence of Gaussian white noise. For the development of this model see Veterbi [8].

Consider the operation of a first order phase-locked loop in the presence of noise (illustrated in Figure 6). The noise, $\eta'(t)$, is Gaussian white noise with a power spectral height of N_0. The differential equation which describes the phase error of the system is

$$\frac{d\phi(t)}{dt} = \frac{d\theta_1(t)}{dt} - K \int_0^t (A \sin(\phi(t-u)) + \eta'(t-u))\delta(u)du \ , \qquad (1\text{-}12)$$

where $\delta(u)$ is the delta or impulse function. Therefore, (1-12) becomes

$$\frac{d\phi(t)}{dt} = \frac{d\theta_1(t)}{dt} - AK \sin(\phi(t)) - K\eta'(t) \ . \qquad (1\text{-}13)$$

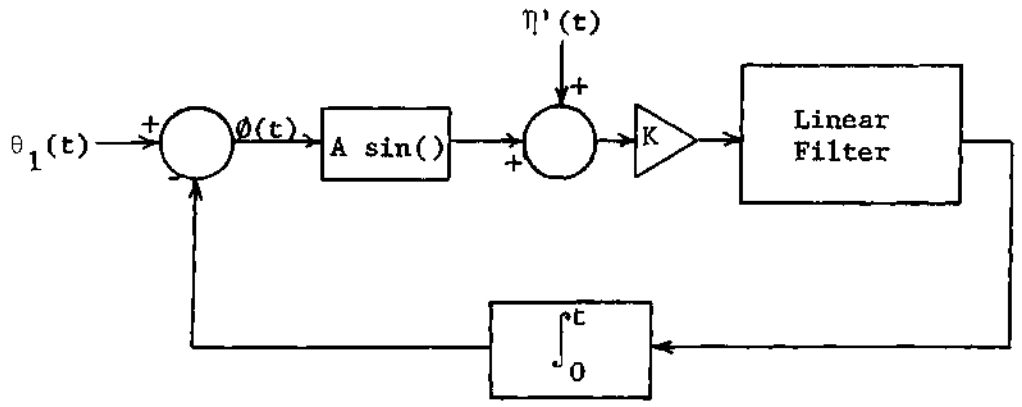


Figure 5. Block Diagram of Phase-Locked Loop in the Presence of Noise

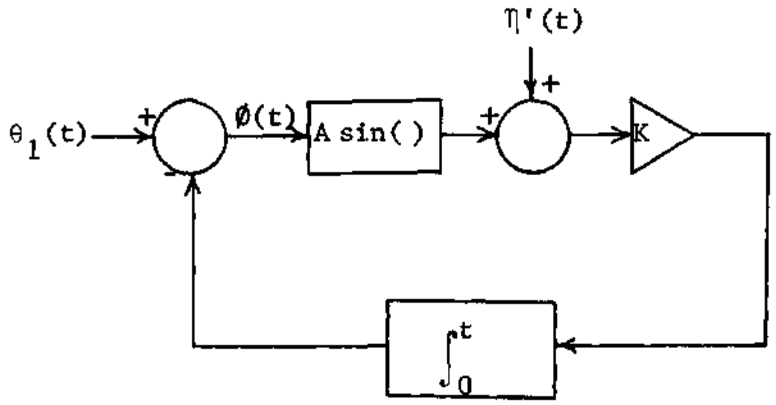


Figure 6. Block Diagram of a First Order Phase-Locked Loop in the Presence of Noise

The Fokker-Planck equation which describes the probability density function of the phase error, $\emptyset(t)$, of the system is

$$\frac{\partial f(\emptyset,t)}{dt} = -\frac{\partial}{\partial\emptyset}\left(C_1(\emptyset,t)\ f(\emptyset,t)\right) + \frac{\partial^2}{\partial\emptyset^2}\left(C_2(\emptyset,t)\ f(\emptyset,t)\right) , \qquad (1\text{-}14)$$

where

$$C_1(\emptyset,t) = \lim_{\Delta t\to 0} \frac{E(\Delta\emptyset)}{\Delta t} = \frac{d\theta_1(t)}{dt} - AK\sin(\emptyset(t)) , \qquad (1\text{-}15)$$

$$C_2(\emptyset,t) = \lim_{\Delta t\to 0} \frac{E(\Delta\emptyset^2)}{\Delta t} = \frac{N_0}{2} .$$

The general form of the Fokker-Planck equation for a second order phase-locked loop in the presence of noise is that of (1-2) with n equal to two.

Numerical Approaches to Solutions of Fokker-Planck Equations

There is a very large amount of literature extending over many years on numerical solutions to partial differential equations (see, for example, bibliographies by Finn [20] and Vichenevetsky [35]). Some methods are quite general in that they apply to more than one type of partial differential equation, while other methods are tailored to a specific type of problem. In this section a summary of the more important numerical methods that might be considered in seeking solutions to Fokker-Planck equations are briefly examined.

Separation of variables [7], which is commonly used to solve simple problems, can be applied to some one-dimensional Fokker-Planck equations. Howe [36] gives a good presentation of the separation of variables technique and shows how it can be implemented on an analog computer.

The Fourier Transformation [3] is another widely known method. However, in general, this method is not applicable to the Fokker-Planck equation. This is because the Fokker-Planck equation, except in the very simplest cases, has variable coefficients.

Iterative methods are another scheme which can be used to solve the Fokker-Planck equation. An iterative method goes through an iterative process which betters an initially guessed solution until it converges to the correct answer. Howe and Hsu [13] developed a formula that reduces the partial differential equation to a coupled set of ordinary differential equations and solves each ordinary differential equation separately and in sequential order. This process is iterated until the answer converges. This technique has also been studied by O'Dowd and Hammond [31].

Mayfield [27] and Lindsey [41] have recently introduced an iterative method which assumes that the problem solution takes a particular form. When the assumed solution is put into the Fokker-Planck equation it reduces the partial differential equation to an integral equation. The integral equation is a Voltera integral equation which can be solved by successive approximations.

Point iterative methods, which are generally applied to elliptic partial differential equations, are also applicable. The better known of these schemes are the Jacobi [14], Gauss-Seidel [14], and successive-overrelaxation [14,15] methods.

In recent years Russian numerical analysts have developed locally one-dimensional methods [5] for solving partial differential equations. These methods separate an n-dimensional problem in such a manner that it is equivalent to n one-dimensional problems.

A large and important class of methods used to solve parabolic partial differential equations is finite difference schemes [1,5,14]. These methods approximate the partial derivatives of the function by finite differences, thereby reducing a partial differential equation to a series of algebraic equations. The most useful finite difference schemes are the one step or single level algorithms. These schemes involve values from only two time levels, t and $t+\Delta t$. The solution at time $t+\Delta t$ is calculated using only the solution at time t. There are two general categories of single level finite difference methods, implicit and explicit.

An explicit formula involves one value at the advanced time $t+\Delta t$. Therefore, to get the solution at $t+\Delta t$, it is only necessary to make N (number of grid points in the space dimension) separate calculations. An implicit method involves more than one grid point at the advanced time $t+\Delta t$. Therefore, it is necessary to solve a set of N simultaneous equations in order to obtain the solution at $t+\Delta t$. One of the better known implicit and explicit finite difference algorithms is discussed in detail in Chapter II.

Alternating direction methods [14,5] are implicit methods which are used in conjunction with certain differencing techniques. The original implicit method is broken into more, but simpler, implicit equations by introducing intermediate variables. The effort required to solve the simpler sets of equations, generally tridiagonal, is much less than that required to solve the original set of equations.

The literature contains a large number of finite difference approximations for partial differential equations. No attempt is made to enumerate these methods. For the most part, these methods were presented without adequate consideration of stability, computer time, and computer storage. Some other general methods, such as the method of lines [34,35], multilevel algorithms [5], and parallel solutions [40] were not discussed because they do not seem to offer any advantages to the type problem being considered.

Comments on the Numerical Methods

Due to practical limitations on computer time and storage, complete solutions (transient and steady-state) to one-dimensional Fokker-Planck equations are rare, and complete solutions to higher-dimensional equations are nonexistent. Almost all previous work treats only the steady-state solution.

Separation of variables can and has been used to solve a one-dimensional Fokker-Planck equation which has coefficients that do not depend on time (the solution is discussed in the next part of this section). It is clear, however, that in the two-dimensional case, the coefficients in the Fokker-Planck equation will in general prevent the variables from being separable.

A method which on the surface seems to offer promise is the locally one-dimensional method. Unfortunately, there is very little work, or experimental results available to use in making a reasonable evaluation of the method. Mitchell [5] is very skeptical of the method. He does, however, point out that this is just his personal conjecture.

Iterative and finite difference methods are the ones commonly used to solve parabolic partial differential equations. In theory these are easily implemented on a computer; however, both require excessive computer time and storage. The iterative methods are particularly demanding on storage since a complete time solution must be stored during the entire simulation. For higher-dimensional equations, where computer storage becomes enormous, it is clear that finite difference schemes offer the best approach to solving the problem.

Recent Solutions to the Fokker-Planck Equation

Almost all previous work on the Fokker-Planck equation for phase-locked loops involves only the steady state solution. Lindsey [26], Viterbi [8], Charles [26], Snyder [23], Holmes [37], and others have worked out steady state results for first order, and in some cases, second order phase-locked loops. Very recently results have begun to be obtained on transient solutions to the Fokker-Planck equation. However, results have been obtained only for the one-dimensional case.

Whitney [15] solved for the complete velocity distribution of a particle in a slightly ionized plasma. However, physical considerations of his problem permitted him to reduce a three-dimensional Fokker-Planck equation to just a few uncoupled one-dimensional Fokker-Planck type equations. The type assumptions that Whitney made are not applicable to other problems.

La Frieda [28] solved for the complete probability density function of the phase error in first order tracking loops. He used the separation of variables technique to get the solution on the modulo 2π state space.

Dominiak and Pickholtz [12] also investigated the phase error in a first order phase-lock loop. They reduced the Fokker-Planck equation to a special case of a one-dimensional heat flow equation which had been investigated at an earlier date by vonNeumann and Richtmyer [22]. The reduced equation was solved by a standard finite difference method. Transient solution curves are presented for several different signal to noise ratios.

CHAPTER II

ESSENTIAL BACKGROUND

The purpose of this chapter is to present material which is drawn on heavily in the development of this thesis. Two general topics are discussed. The first is a finite difference approach to solving parabolic partial differential equations. The second topic is techniques for polynomial interpolation between discrete functional values.

The emphasis of this chapter is on presenting the ideas and showing how they are used and not on the mathematical developments.

Finite-Difference Equations

As noted in Chapter I, the finite-difference methods are among the most successful approaches to numerical solution of partial differential equations. The use of finite differences reduces the partial differential equation to a set of algebraic equations. Suppose the partial differential equation has n+1 independent variables; n state variables, x_i, and time, t. The finite difference scheme divides the portion of the space over which a solution is sought ($a_1 \le x_1 \le b_1$, $a_2 \le x_2 \le b_2$, . . ., $a_n \le x_n \le b_n$, $0 \le t \le T$) into discrete points by placing a grid, or lattice, on it. Information about the solution is retained only at the preselected grid points. A finite difference algorithm involves approximating the function values and their derivatives, as required by the partial differential equation, by finite differences. Values for the solution are only obtained at the grid points. For a good reference see Ralston and Wilf [1].

The finite difference formulation will be illustrated with a one-dimensional problem. Consider the general, linear, one-dimensional, second order, parabolic partial differential equation

$$L(U(x,t)) = U_t(x,t) - a(x,t)\, U_{xx}(x,t) - 2b(x,t)\, U_x(x,t) + c(x,t)\, U(x,t) - d(x,t) = 0 \ , \tag{2-1}$$

where $a(x,t) > 0$ and L is a differential operator. In (2-1) partial derivatives are denoted by the subscripts. For example, the second partial derivative of $U(x,t)$ with respect to x is denoted by $U_{xx}(x,t)$. Similarly, the first partial derivative of $U(x,t)$ with respect to t is represented by $U_t(x,t)$.

The solution of (2-1) is sought on the semi-infinite strip

$$S{:}(A \le x \le B,\ t > 0) \ . \tag{2-2}$$

If the terms of (2-1) are analytic functions on the region S, the solution of (2-1) is uniquely determined by specifying initial and boundary conditions [45]. For instance, let

$$\begin{aligned} U(x,0) &= g(x),\ A < x < B \\ U(A,t) &= h_1(t),\ t > 0 \\ U(B,t) &= h_2(t),\ t > 0 \ . \end{aligned} \tag{2-3}$$

A grid defined by

$$S_{\Delta x,\Delta t}:(x_j = A + j\Delta x,\ j = 0,1,2,. . .,J;\ t = n\Delta t, \qquad (2\text{-}4)$$
$$n = 0,1,2,. . .,N)\ ,$$

where $\Delta x = \frac{B-A}{J}$, is placed on the strip S. This grid is illustrated in Figure 7. It is desired to solve for $U(x_j,t_n)$ at each point on the grid. This is accomplished by solving an appropriate set of finite difference equations used to approximate the partial differential equation.

It is convenient to denote the dependent variable at the grid points as

$$U(A + j\Delta x,\ n\Delta t) = U_j^n\ , \qquad (2\text{-}5)$$

and the approximation to the true solution as

$$V(A + j\Delta x,\ n\Delta t) = V_j^n\ . \qquad (2\text{-}6)$$

Note that in using this notation the subscript gives the space variable index and the superscript gives the time variable index.

Finite difference equations are classified as either implicit or explicit equations. Each explicit equation can be solved directly and easily. This is not the case for the implicit difference equations. At a given time the implicit finite difference equations are a coupled set of linear equations which have to be solved simultaneously. Each method has certain advantages and disadvantages.

In order to develop the finite difference method as a general approach which contains both the implicit and explicit methods, it is necessary to have values of U(x,t) and its space derivatives at intermediate values of time which do not correspond to grid points. This is accom-

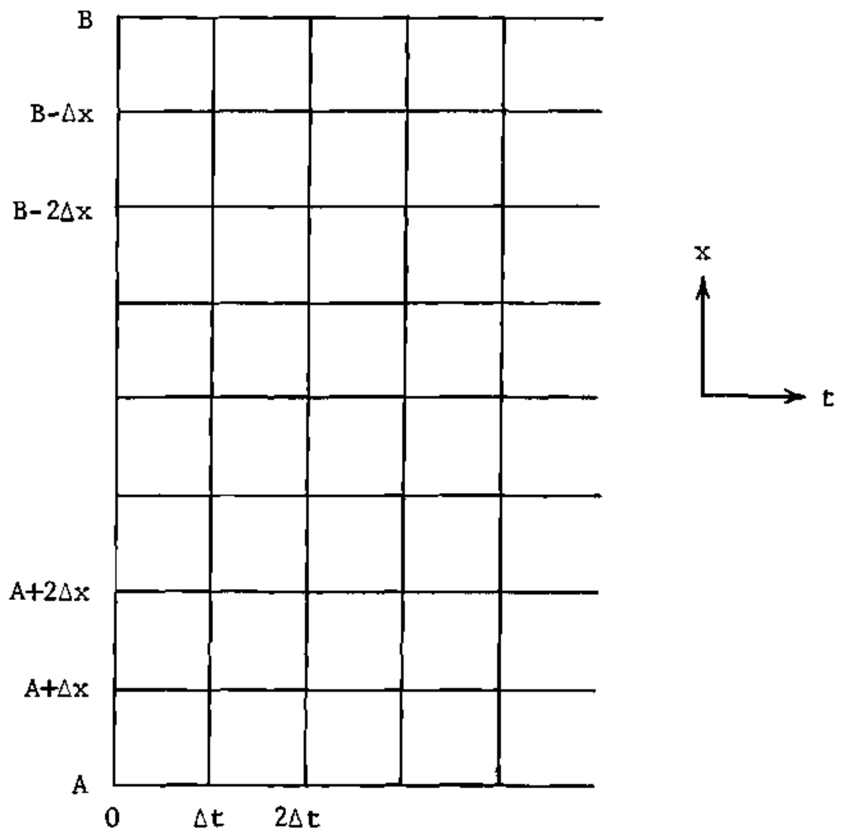


Figure 7. Typical Grid for a One-Dimensional Problem

plished using linear interpolation in time. No such intermediate values are required in the space dimension.

Using linear interpolation, the approximation to U(x,t) at

$$x = x_j = A + j\Delta x \ , \qquad (2\text{-}7)$$

$$t = t_{n+\theta} = (n+\theta)\Delta t \ ,$$

is given by

$$V(A + j\Delta x,\ (n+\theta)\Delta t) = V_j^{n+\theta} = \theta V_j^{n+1} + (1-\theta)V_j^n \ , \qquad (2\text{-}8)$$

where

$$0 \le \theta \le 1 \ .$$

Centered difference approximations are used to represent the derivatives of the function with respect to the space variable. The time derivative is represented by a forward difference approximation. The resulting finite difference approximations are

$$U_x(x_j,\ t_{n+\theta}) = \frac{V_{j+1}^{n+\theta} - V_{j-1}^{n+\theta}}{2\Delta x} \ , \qquad (2\text{-}9)$$

$$U_{xx}(x_j,\ t_{n+\theta}) = \frac{V_{j+1}^{n+\theta} - 2V_j^{n+\theta} + V_{j-1}^{n+\theta}}{\Delta x^2} \ ,$$

and

$$U_t(x_j,\ t_{n+\theta}) = \frac{V_j^{n+1} - V_j^n}{\Delta t} \ .$$

These approximations are put into (2-1) in order to obtain the finite difference equation which approximates the partial differential equation.

Combining (2-1), (2-9), and (2-8) the resulting difference equation at a typical grid point is

$$V_{j+1}^{n+1}\left(-\theta\,\frac{a_j^{n+\theta}\Delta t}{\Delta x^2}-\theta\,\frac{b_j^{n+\theta}\Delta t}{\Delta x}\right)+V_j^{n+1}\left(1+\theta\,\frac{2a_j^{n+\theta}\Delta t}{\Delta x^2}+\theta\,c_j^{n+\theta}\Delta t\right)\qquad(2\text{-}10)$$

$$+\,V_{j-1}^{n+1}\left(-\theta\,\frac{a_j^{n+\theta}\Delta t}{\Delta x^2}+\theta\,\frac{b_j^{n+\theta}\Delta t}{\Delta x}\right)=V_{j+1}^{n}\left((1-\theta)\,\frac{a_j^{n+\theta}\Delta t}{\Delta x^2}+(1-\theta)\times\right.$$

$$\left.\frac{b_j^{n+\theta}\Delta t}{\Delta x}\right)+V_j^{n}\left(1-(1-\theta)\,\frac{2a_j^{n+\theta}\Delta t}{\Delta x^2}-(1-\theta)\,c_j^{n+\theta}\Delta t\right)+V_{j-1}^{n}\times$$

$$\left((1-\theta)\,\frac{a_j^{n+\theta}\Delta t}{\Delta x^2}-(1-\theta)\,\frac{b_j^{n+\theta}\Delta t}{\Delta x}\right)+\Delta t\;d_j^{n+\theta}\;.$$

The initial and boundary conditions, (2-3), now become

$$V_j^0=g(x_j)\;,\;0\le j\le J\qquad(2\text{-}11)$$

$$V_0^n=h_1(t_n)\;,\;n>0$$

$$V_J^n=h_2(t_n)\;,\;n>0\;.$$

The solution to (2-1) is approximated by the solution to the set of difference equations defined by (2-10). There is one finite difference equation which must be solved for each grid point on the lattice (see Figure 7).

The initial condition, (2-11), gives the solution of (2-10) for n=0 (corresponding to t=0). The solution for n=1 (corresponding to t=Δt) can then be obtained using (2-10). In a like manner the solutions at all the grid points for n=2 can be obtained using (2-10) and the solutions at n=1. In general (2-10) relates the solutions of the equations at n+1 to the solution at n.

If $\theta=0$, (2-10) is called an explicit finite difference equation. For this case (2-10) becomes

$$V_j^{n+1} = V_{j+1}^n \left(\frac{a_j^n \Delta t}{\Delta x^2} + \frac{b_j^n \Delta t}{\Delta x}\right) + V_j^n \left(1 - \frac{2a_j^n \Delta t}{\Delta x^2} - c_j^n \Delta t\right) \qquad (2\text{-}12)$$

$$+ V_{j-1}^n \left(\frac{a_j^n \Delta t}{\Delta x^2} - \frac{b_j^n \Delta t}{\Delta x}\right) + d_j^n \Delta t \ .$$

Note that (2-12) expresses the solution at any point along the grid corresponding to n+1 explicitly in terms of solutions along the grid line at n. Therefore, given the solution values at n, V_j^{n+1}, for j=1,2, . . ., J-1, can be determined directly.

In order to solve for the value at j on the n+1 line, (2-12) shows that three solution values on the line at n are required; j-1, j, and j+1. The relationship of these four values is illustrated in Figure 8. The values of the solution at the points marked by the boxes are necessary in order to compute the solution at the point marked by the circle. In this manner a solution for all j=1,2, . . ., J-1 on the n+1 line can be obtained from the complete solution (j=1,2, . . ., J-1) along the n line and the boundary conditions. Therefore, starting with the initial conditions, the explicit finite difference scheme provides a direct method of progressing through a sequence of values of j and n in order to obtain

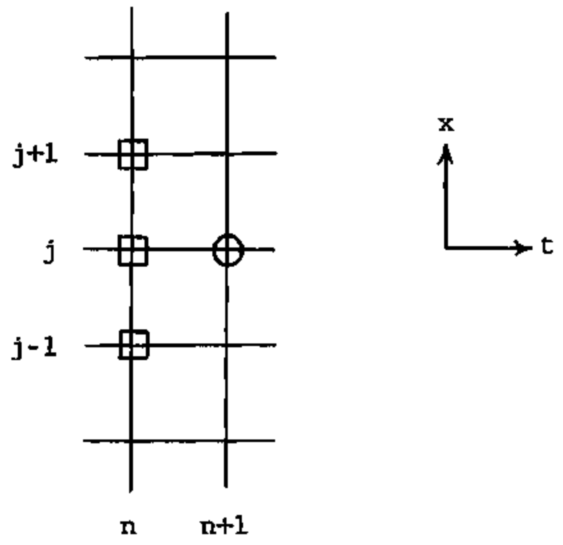


Figure 8. Relationship Between Values in an Explicit Finite Difference Scheme

a complete solution to (2-1).

If $\theta \neq 0$ in (2-10), the solution for a typical point on line n+1 is not related in a simple manner to solutions on the line n, and (2-10) cannot be solved directly. If, however, a set of equations is developed for each V_j^{n+1} ($j=1,2, \ldots, J-1$), the result is a coupled system of linear equations which can be solved simultaneously. If $\theta \neq 0$ the algorithm is called an implicit finite difference equation.

Before finite difference equations of either type can be solved, suitable step sizes (Δx and Δt) must be selected. There are a number of considerations which enter into such selections. The step sizes and the ratio of the step sizes determine the accuracy of the method and whether or not the algorithm is "stable," "convergent," and "consistent" [1]. These parameters, which characterize the algorithm, are defined and discussed below.

A difference scheme is consistent if the difference equations do actually approximate the partial differential equation. Stated more precisely, the difference scheme is consistent with the partial differential equation if

$$\lim_{\Delta t,\Delta x \to 0} \left| L(U(x,t)) - L_{\Delta x,\Delta t}(U(x,t)) \right| = 0 \ . \tag{2-13}$$

where $L_{\Delta x,\Delta t}(U(x,t))$ is the differential operator defined by (2-10).

The difference scheme is convergent if

$$\lim_{\Delta x,\Delta t \to 0} \left| U(x_j,t_n) - V_j^n \right| = 0 \ . \tag{2-14}$$

Convergence means that, for sufficiently small step sizes, the numerical solution of the difference equations at each grid point is a close approximation to the exact solution of the partial differential equation at the corresponding grid points.

The difference equations are stable if small errors introduced in the solution remain bounded as computations progress to other points in the grid.

Keller presents a rule [1], called the maximum principle, that shows how to select step sizes. For (2-1), the maximum principle states that, if Δx and Δt are chosen so that

$$1 + \theta \Delta t \; c(x,t) > 0 \; , \tag{2-15}$$

$$a(x,t) - \Delta x |b(x,t)| \geq 0 \; ,$$

and

$$1 - (1-\theta) \left(2 \frac{\Delta t}{\Delta x^2} a(x,t) + \Delta t \; c(x,t) \right) \geq 0 \; ,$$

then the algorithm, (2-10) is consistent, convergent, and stable.

It should be pointed out that the maximum principle is not an if and only if statement. That is, if the step sizes are selected in accordance with the maximum principle, the algorithm will be consistent, convergent, and stable. If, however, the step sizes are selected by some other criterion which does not satisfy (2-15), the maximum principle cannot be used to draw any conclusions. The choice of step sizes may or may not cause the algorithm to be convergent, consistent, and stable.

A common measure of accuracy of a numerical scheme is the truncation error of the algorithm. This is a useful measure since solution errors go to zero as the truncation error goes to zero. The truncation error is defined as

$$L(U) - L_{\Delta x,\Delta t}(U) = \tau \; . \tag{2-16}$$

For the terms in (2-1) and (2-10) the truncation errors are

$$U_x(x_j,t_{n+\theta}) - \frac{U_{j+1}^{n+\theta} - U_{j-1}^{n+\theta}}{2\Delta X} = \frac{\Delta x^2}{6} U_{xxx}(\xi_1,\delta_1) \tag{2-17}$$

$$+ \frac{\theta(1-\theta)}{2} \Delta t^2 U_{xtt}(\xi_1,\delta_1) = \tau_1 \; ,$$

$$U_{xx}(x_j,t_{n+\theta}) - \frac{U_{j+1}^{n+\theta} - 2U_j^{n+\theta} + U_{j-1}^{n+\theta}}{\Delta X^2} = \frac{\Delta x^2}{12} U_{xxxx}(\xi_2,\delta_2)$$

$$+ \frac{\theta(1-\theta)}{2} \Delta t^2 \; U_{xxtt}(\xi_2,\delta_2) = \tau_2 \; ,$$

$$U_t(x_j,t_{n+\theta}) - \frac{U_j^{n+1} - U_j^n}{\Delta t} = \frac{1-2\theta}{2} \Delta t \; U_{tt}(\xi_3,\delta_3) + \frac{1-3\theta+3\theta^2}{6} \times$$

$$\Delta t^2 \; U_{ttt}(\xi_3,\delta_3) = \tau_3 \; ,$$

where

$$X_{j-1} \le \xi_i \le X_{j+1} \; , \; i = 1,2,3$$

$$t_n \le \delta_j \le t_{n+1} \; , \; j = 1,2,3.$$

Inserting the truncation errors for each term in (2-10) the total truncation error becomes

$$\tau_T = \tau_3 - a\tau_2 + 2b\tau_1 \; . \tag{2-18}$$

As can be seen from (2-17), in general this truncation error is of order Δx^2 and Δt, or symbolically

$$\tau_T = 0(\Delta x^2) + 0(\Delta t) \ . \tag{2-19}$$

If, however, θ is equal to one-half, the truncation error becomes

$$\tau_T = 0(\Delta x^2) + 0(\Delta t^2) \ . \tag{2-20}$$

For this special case the algorithm is known as the Crank-Nicholson method [14].

General Comments on Implicit and Explicit Methods

For explicit methods, $\theta=0$, the computations are quite simple and easily performed. However, if the maximum principle is used to select Δx and Δt, the total number of calculations will be large. This in turn requires a large amount of computer time and storage.

If $\theta \neq 0$ the finite difference equations are implicit. To advance the solution one step in time for this case requires the solution of a coupled set of J-1 equations. The complete solution for one step in time thus requires more work than a complete solution to advance one step forward in time using an explicit method. However, implicit methods have some very definite advantages.

For the completely implicit difference scheme, $\theta=1$, (2-15) places no restraints between Δt and Δx and requires only that

$$1 + \Delta t \ c(x,t) > 0 \ . \tag{2-21}$$

Therefore, the total computations required to reach some time t can be made less than in the explicit case by choosing Δt sufficiently large. However, as Δt is allowed to increase, the accuracy of the numerical scheme decreases.

In the Crank-Nicholson method ($\theta = 1/2$), a higher degree of accuracy appears to be obtained ($O(\Delta x^2) + O(\Delta t^2)$). For this method, equation (2-15) restricts Δt to be no larger than twice the allowable Δt for the corresponding explicit method.

The trade offs between the implicit and explicit algorithms involve the complexity in obtaining solutions and the time step size required in order to have a stable scheme. For comparable accuracy, computer time, and computer storage, it is not really clear which method should be used to solve a one-dimensional partial differential equation. In general, either method can be used to satisfactorily simulate one-dimensional problems.

However, for higher-dimensional parabolic partial differential equations (two or more space variables), the implicit equations become very difficult to handle. This is because the number of algebraic equations which must be solved simultaneously is very large. For higher-dimensional problems the explicit scheme is preferable.

Polynomial Interpolation

In the development of the numerical method used in this thesis, polynomial interpolation [24,25] is used in conjunction with an explicit finite difference scheme to form a new algorithm. In this section the mechanics of the polynomial interpolation scheme to be used are examined.

The basic idea is to pass a polynomial through a known set of discrete values of a function and use the value of this polynomial at other points to approximate the value of the function there. For example, if $F(0) = 1$ and $F(2) = 11$, the linear approximating polynomial

$$F(x) = 5x + 1 \tag{2-22}$$

can be fitted to the known values. The value $F(0.3)$ can then be approximated by

$$F(0.3) = 5(0.3) + 1 = 2.5 \; . \tag{2-23}$$

In general if n+1 values of a function are known at a corresponding set of argument values, an n^{th} order polynomial can be fitted to these values. This polynomial can then be used to approximate the function at any intermediate values of its argument.

Isaacson and Keller [25] show that for any n+1 values the n^{th} order interpolation polynomial exists and is unique. There are many methods of computing the polynomial: Lagrange method, Newton's method, iterative linear interpolation, forward and centered difference schemes, etc. Each has its own particular advantages and characteristics. For instance, the iterative linear interpolation methods are a class of methods for generating successively higher order interpolation polynomials. That is, if another point of interpolation is added, then the new higher degree polynomial is easily computed. However, regardless of how the n^{th} order polynomial is fitted to the n+1 data points, the result is the same for all methods.

The method used in this thesis is an iterative linear method that uses a forward difference scheme to fit an n^{th} order polynomial to n+1 equally spaced data points. The value of the function is known at the points

$$x_j = x_0 + j\Delta x, \quad j = 0,1, \ldots, n. \tag{2-24}$$

Let

$$x = x_0 + h\Delta x \ , \tag{2-25}$$

where h is a continuous variable such that

$$0 \le h \le n \ . \tag{2-26}$$

It is convenient to introduce the notation

$$\begin{aligned} \pi_0(h) &= h \ , \qquad (2\text{-}27) \\ \pi_1(h) &= h(h-1) \ , \\ \pi_n(h) &= h(h-1) \ldots, (h-n) \ , \ n = 2,3 \ . \end{aligned}$$

The function $\pi_n(h)$ is a polynomial of degree n+1 and is generally called the $(n+1)^{st}$ factorial polynomial.

Forward differences are to be used in the approximation. These differences are

$$\begin{aligned} \Delta F(x_0) &= F(x_1) - F(x_0) \ , \qquad (2\text{-}28) \\ \Delta^2 F(x_0) &= F(x_2) - 2F(x_1) + F(x_0) \ , \\ \Delta^3 F(x_0) &= F(x_3) - 3F(x_2) + 3F(x_1) - F(x_0) \ , \text{ etc.} \end{aligned}$$

In general

$$\Delta^n F(x_0) = \Delta^{n-1} F(x_1) - \Delta^{n-1} F(x_0) \ . \tag{2-29}$$

The n^th order polynomial which approximates the function over the range of the data, $(x_0, x_0+n\Delta x)$, is

$$P_n(x_0 + h\Delta x) = F(x_0) + \frac{\pi_0(h)}{1!} \Delta F(x_0) + \frac{\pi_1(h)}{2!} \Delta^2 F(x_0) \tag{2-30}$$

$$+ \ . \ . \ . + \frac{\pi_{n-1}(h)}{h!} \Delta^n F(x_0) \ ,$$

$$0 \le h \le n \ .$$

The error or remainder term associated with this polynomial approximation is

$$R_n(x_0 + h\Delta x) = \frac{\pi_n(h) \ \Delta x^{n+1}}{(n+1)!} F^{(n+1)}(\xi) \ , \tag{2-31}$$

$$x_0 < \xi < x_n \ .$$

The approximation (2-30) usually gives very good results near the center of the interval over which the function is approximated. The approximation becomes progressively less accurate away from the center of the interpolating region. This is fairly obvious if the function $\pi_n(h)$ is plotted.

One interesting and not so obvious fact should be pointed out. It is not generally true that higher degree interpolation polynomials yield more accurate approximations. In fact, for equidistant points of interpolation, such as considered here, relatively low order polynomials give the most accurate results [25].

CHAPTER III

DEVELOPMENT OF THE MODIFIED ALGORITHM

Parabolic partial differential equations solved by the methods discussed in the previous chapter require large amounts of computer time and storage. This is particularly true for equations with two or more space dimensions. With a finite difference scheme the function value has to be calculated and stored for every grid point on the $t_n (n=1,2,. . .)$ line. Since most grids must contain a very large number of points in order to give the required accuracy and stability, a large number of computer operations and a large amount of computer storage are required. The amount of computer time and storage required to solve multi-dimensional problems is so large that very few solutions are attempted.

A modified algorithm for solving parabolic partial differential equations is developed in this chapter. The modified method requires much less computer time and storage while maintaining the same degree of accuracy as standard finite difference schemes.

The modified algorithm combines an explicit finite difference equation and polynomial interpolation. To solve a partial differential equation a grid is set up in the same manner as for an explicit finite difference scheme. The modified method differs from the conventional one in that on every line of the grid, for fixed n, the solution is calculated and stored at only one out of every P points. The non-stored values which are necessary in order to advance the solution to line n+1 are approximated by fitting polynomials to the stored values. The net effect is an algorithm which stores and computes $(1/P)^{th}$ of the amount of data as the original algorithm and yet maintains effectively the same grid size as the original method.

The modified method is first developed for one-dimensional equations. The algorithm is then extended to two-dimensional problems of two different types.

One-Dimensional Equation

To introduce the modified algorithm, consider the one-dimensional parabolic partial differential equation

$$F_t(x,t) = a(x,t)F_{xx}(x,t) + 2b(x,t)F_x(x,t) \ , \tag{3-1}$$

with the initial and boundary conditions

$$\begin{aligned} F(x,0) &= g(x), \quad A < x < B \qquad (3\text{-}2) \\ F(A,t) &= h_1(t), \quad t > 0 \\ F(B,t) &= h_2(t), \quad t > 0 \ . \end{aligned}$$

Although (3-1) is more specialized than (2-1), discussed in the previous chapter, it is sufficiently general for present purposes.

The finite difference equation which approximates (3-1) is*

*The notation used in the remainder of the thesis is a modification of that used in Chapter II. In all further work the discrete variable, F_i^n, is used to represent either the true value of the function or the approximate value of the function. Which is meant is obvious from its use. For the most part F_i^n is the approximate value of the function.

$$F_i^{n+1} = F_{i+1}^n \left(a(x_i,t_n) \frac{\Delta t}{\Delta x^2} + b(x_i,t_n) \frac{\Delta t}{\Delta x}\right) \tag{3-3}$$

$$+ F_i^n \left(1 - 2a(x_i,t_n) \frac{\Delta t}{\Delta x^2}\right) + F_{i-1}^n \left(a(x_i,t_n) \frac{\Delta t}{\Delta x^2} - b(x_i,t_n) \frac{\Delta t}{\Delta x}\right) .$$

The step sizes (Δx and Δt) are chosen to satisfy the maximum principle.

The modified method computes and stores the solution values at every P^{th} grid point, where P is an integer, rather than at every grid point as in conventional methods. This is illustrated in Figure 9 with P equal to five.

Consider the typical step of progressing from the line n to the line n+1. As a result of the previous step the computed values of $F(x_i,t_n)$ are stored at every P^{th} grid point along the line n. It is desired to compute $F(x_i,t_{n+1})$ at every P^{th} point along the line n+1. In order to use (3-3) to perform the calculations, additional values of $F(x,t_n)$ are required. These values are obtained by using polynomial interpolation between the stored values. Figure 10 illustrates the information required to compute the answer at every P^{th} point when P is equal to five.

Only one third of the values necessary on line n in order to compute solutions at every P^{th} grid point on line n+1 are stored. The additional required values (represented by the boxes in Figure 10) are obtained by fitting polynomials to the stored values (represented by the x's in Figure 10).

Figures 11 and 12 illustrate how second and third order polynomial interpolations are used to regenerate the required data. These orders of

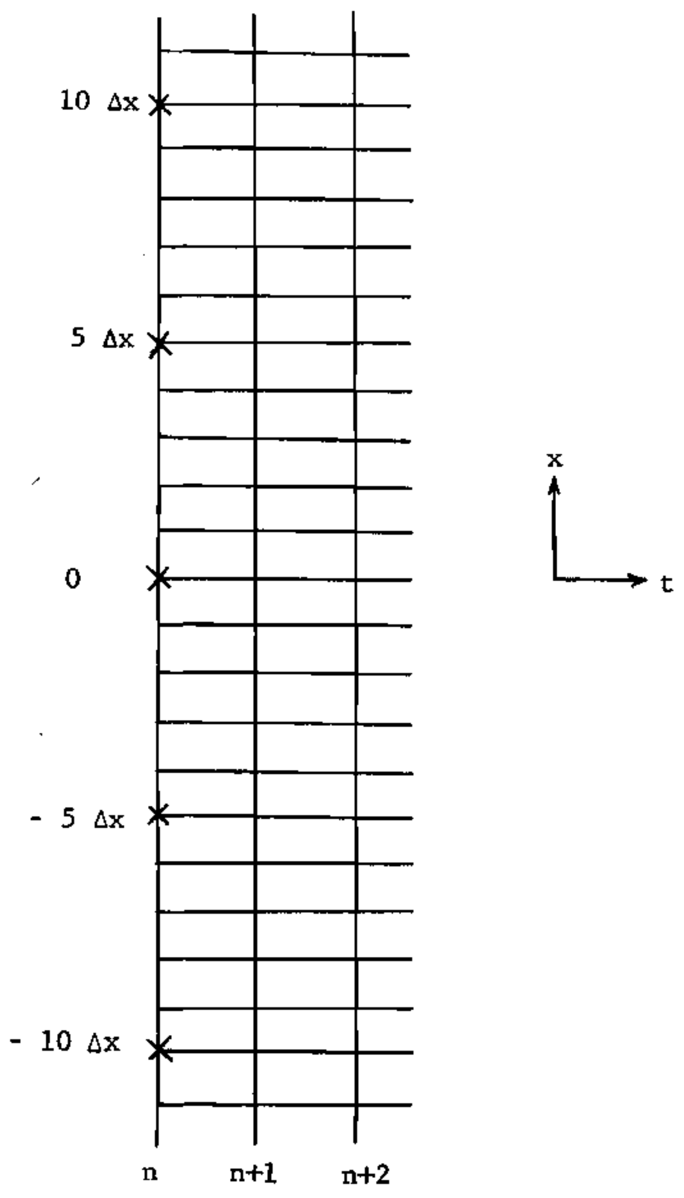


Figure 9. Storage Scheme for the Modified Algorithm with P=5

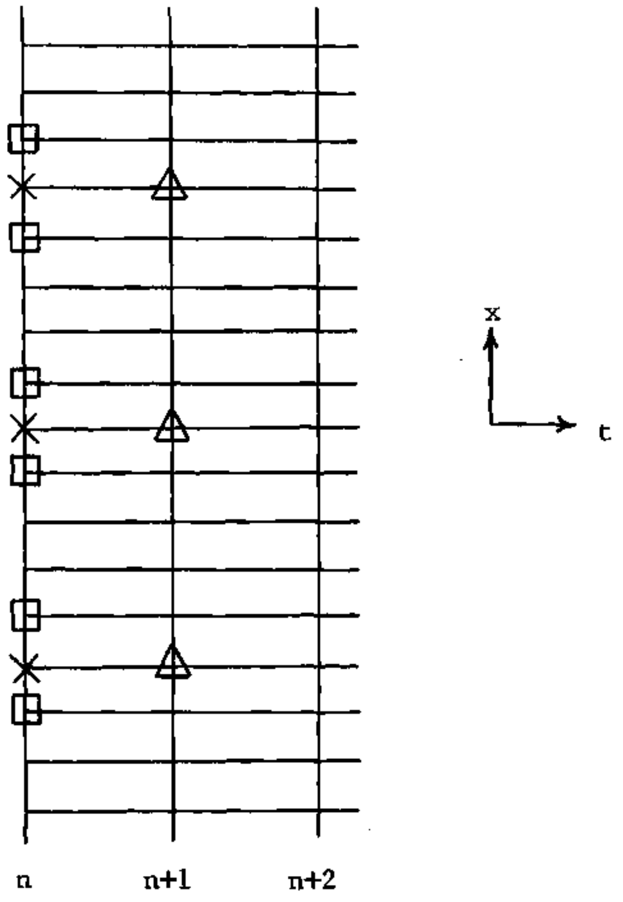


Figure 10. Information Necessary at $t=n\Delta t$ in Order to Compute the Solution at Every P^{th} Grid Point at $t=(n+1)\Delta t$

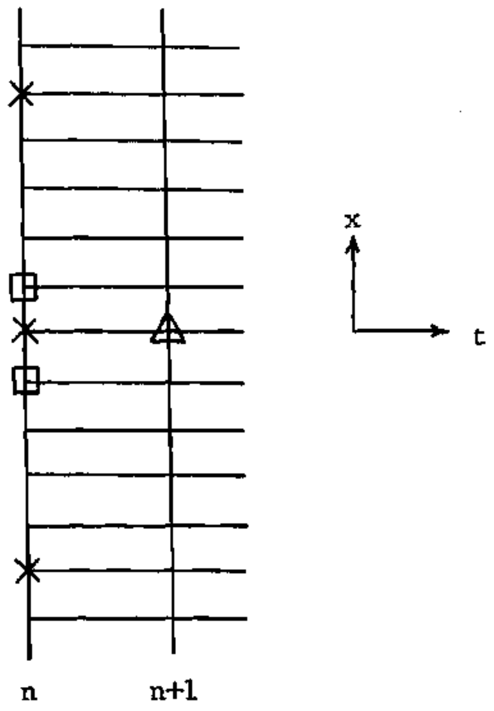


Figure 11. Second Order Polynomial Approximation

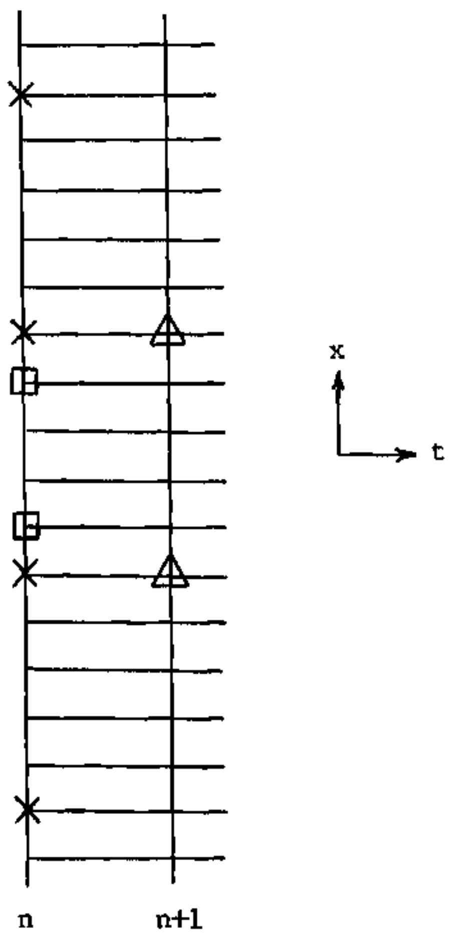


Figure 12. Third Order Polynomial Approximation

polynomial approximations are studied in a later chapter to determine which gives the best results.

As in the conventional case, the new method can be used to compute the complete solution, starting with the initial conditions at n=0 and progressing to n=N.

Convergence, Stability, and Consistency

As was pointed out in Chapter II, in order for an algorithm to be useful it must be convergent, consistent, and stable. These properties of the modified algorithm are investigated below.

Starting with the true solutions at the stored points along line n, the properties of the modified algorithm are investigated as computations progress. Figure 13 illustrates the procedure for P equal to five. The stored values on line n (represented by the x's in Figure 13) are assumed known exactly.

Consistency is the first property examined. Consistency means that the truncation error of the differential operator must go to zero as the step sizes go to zero. That is

$$\lim_{\Delta x \to 0, \Delta t \to 0} \left| L(F) - \hat{L}_{\Delta x, \Delta t}(F) \right| = 0 \ , \qquad (3\text{-}4)$$

where $\hat{L}_{\Delta x, \Delta t}(F)$ is the modified differential operator.

The modified algorithm must first compute the additional information required on line n (represented by the circles in Figure 13) in order to use (3-3) to calculate solution values on line n+1 (represented by the triangle in Figure 13). The additional information is obtained by fitting

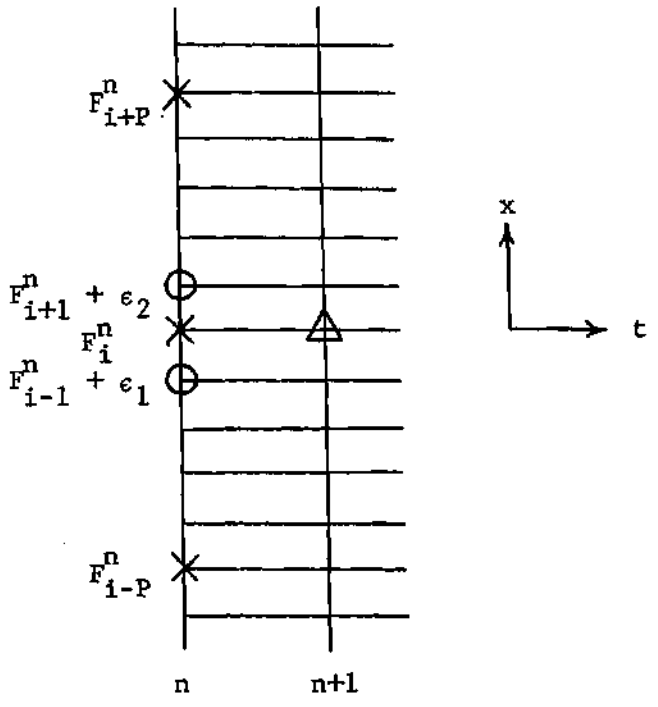


Figure 13. Consistency, Convergence, and Stability Study for the Modified Algorithm

polynomials to the stored data. Since initially it is assumed that the stored data are known exactly, the computed function values can be expressed as

$$\hat{F}^n_{i+1} = F^n_{i+1} + \epsilon_2 \ , \qquad (3\text{-}5)$$

$$\hat{F}^n_{i-1} = F^n_{i-1} + \epsilon_1 \ ,$$

where F^n_{i+1} and F^n_{i-1} are the true values of the function and ϵ_2 and ϵ_1 are the polynomial interpolation errors. These error terms, given by (2-31) are

$$\epsilon_i = O(\Delta x^{Q+1}) \ , \qquad (3\text{-}6)$$

where Q is the order of polynomial interpolation used.

The modified finite difference operator (equation) for (3-1) now becomes

$$\hat{L}_{\Delta x,\Delta t} = \frac{F^{n+1}_i - F^n_i}{\Delta t} - a^n_i \frac{F^n_{i+1} + \epsilon_2 - 2F^n_i + F^n_{i-1} + \epsilon_1}{\Delta x^2} \qquad (3\text{-}7)$$

$$+ 2b^n_i \frac{F^n_{i+1} + \epsilon_2 - F^n_{i-1} - \epsilon_1}{2\Delta x} = L_{\Delta x,\Delta t} - a^n_i \frac{\epsilon_2 + \epsilon_1}{\Delta x^2} - b^n_i \frac{\epsilon_2 - \epsilon_1}{\Delta x} = 0 \ ,$$

where $L_{\Delta x,\Delta t}$ is the original explicit finite difference operator, (3-3). Therefore, the truncation error of the modified algorithm is

$$\hat{\tau} = \tau + a^n_i \frac{\epsilon_2 + \epsilon_1}{\Delta x^2} + b^n_i \frac{\epsilon_2 - \epsilon_1}{\Delta x} \ , \qquad (3\text{-}8)$$

where τ is the truncation error of the explicit finite difference method.

The truncation error of the explicit finite difference scheme is

$$\tau = 0(\Delta x^2) \ . \tag{3-9}$$

Using (3-6), (3-8), and (3-9), the truncation error of the modified algorithm can be expressed as

$$\hat{\tau} = 0(\Delta x^2) + 0(\Delta x^{Q-1}) + 0(\Delta x^Q) \ . \tag{3-10}$$

Therefore, if a second or higher order interpolating polynomial is used, the modified algorithm is consistent.

If an even order polynomial interpolation is used, the second term on the right side of (3-10) becomes $0(\Delta x^Q)$. This is demonstrated for second order interpolation, Q = 2. The term on the right hand side of (3-8) which yields the $0(\Delta x^{Q-1})$ term in (3-10) is

$$a_i^n \frac{\epsilon_2 + \epsilon_1}{\Delta x^2} = \frac{a_i^n \ K \ \Delta x^3}{\Delta x^2} \ (F^{(3)}(\epsilon) - F^{(3)}(\epsilon')) \ , \tag{3-11}$$

where K is a constant (K = 4 for P = 5) and

$$x_{i-P} < \epsilon < x_{i+P} \ , \tag{3-12}$$
$$x_{i-P} < \epsilon' < x_{i+P} \ .$$

By the mean value theorem (assuming the fourth derivative exists and is continuous)

$$F^{(3)}(\epsilon) - F^{(3)}(\epsilon') = (\epsilon - \epsilon')\, F^{(4)}(\epsilon'') \,, \qquad (3\text{-}13)$$
$$\epsilon' < \epsilon'' < \epsilon \,,$$
$$|\epsilon - \epsilon'| < 2P\Delta x \,.$$

Therefore

$$\left| a_i^n \frac{\epsilon_2 + \epsilon_1}{\Delta x^2} \right| < |a_i^n K' \Delta x^2 F^{(4)}(\epsilon'')| = O(\Delta x^Q) \,. \qquad (3\text{-}14)$$

Thus, from (3-10) and (3-14) it can be concluded that the modified algorithm has the same truncation error as the original explicit finite difference method if $Q \geq 2$.

If $Q \geq 2$ Keller's proof [1] of stability and convergence can be used directly, with the substitution of the modified truncation error, to demonstrate the stability and convergence of the modified algorithm.

A study is made in Chapter IV to see which order interpolating polynomial gives the best results.

Two-Dimensional Equation

In this section the method just developed for the numerical solution of one-dimensional problems is extended to two-dimensional parabolic partial differential equations. First, the algorithm is developed for standard two-dimensional equations. Then the algorithm is extended to a more complicated two-dimensional form which is useful in solving two-dimensional Fokker-Planck equations.

Standard Two-Dimensional Parabolic Equations

Consider the two-dimensional parabolic partial differential equation

$$F_t(x_1,x_2,t) - a_1(x_1,x_2,t)\, F_{x_1x_1}(x_1,x_2,t) \tag{3-15}$$

$$-a_2(x_1,x_2,t)\, F_{x_2x_2}(x_1,x_2,t) - 2b_1(x_1,x_2,t)\, F_{x_1}(x_1,x_2,t)$$

$$- 2b_2(x_1,x_2,t)\, F_{x_2}(x_1,x_2,t) + c(x_1,x_2,t)\, F(x_1,x_2,t) = d(x_1,x_2,t) \;,$$

$$a_1(x_1,x_2,t) > 0 \;,$$

and

$$a_2(x_1,x_2,t) > 0 \;.$$

The initial boundary conditions are given by

$$F(x_1,x_2,0) = g(x_1,x_2) \;, \tag{3-16}$$

$$A < x_1 < B \quad \text{and} \quad C < x_2 < D$$

$$F(A,x_2,t) = h_1(x_2,t) \;, \qquad t > 0$$

$$F(B,x_2,t) = h_2(x_2,t) \;, \qquad t > 0$$

$$F(x_1,C,t) = h_3(x_1,t) \;, \qquad t > 0$$

$$F(x_1,D,t) = h_4(x_1,t) \;, \qquad t > 0 \;.$$

An extension of the method developed in the previous section for one-dimensional problems is employed to solve (3-15).

In essence the procedure is as follows. The explicit finite difference equation approximating (3-15) is formed and a grid in the variables x_1, x_2, and t is set up. Appropriate step sizes are chosen. A procedure analogous to that for one-dimensional equations is developed which uses only $(1/P)^2$ of the data which would be required by the standard finite difference method ($(1/P)^{th}$ in each space dimension). As in the one-dimensional case, the additional required data are obtained by polynomial interpolation.

To develop the procedure for two-dimensions, let

$$F^n_{ij} = F(A+i\Delta x_1,\ C+j\Delta x_2,\ n\Delta t) \ . \tag{3-17}$$

In a similar manner define

$$\begin{aligned} a^n_{1_{ij}} &= a_1(A+i\Delta x_1,\ C+j\Delta x_2,\ n\Delta t) \ , \qquad (3\text{-}18) \\ a^n_{2_{ij}} &= a_2(A+i\Delta x_1,\ C+j\Delta x_2,\ n\Delta t) \ , \\ b^n_{1_{ij}} &= b_1(A+i\Delta x_1,\ C+j\Delta x_2,\ n\Delta t) \ , \\ b^n_{2_{ij}} &= b_2(A+i\Delta x_1,\ C+j\Delta x_2,\ n\Delta t) \ , \\ c^n_{1_{ij}} &= c_1(A+i\Delta x_1,\ C+j\Delta x_2,\ n\Delta t) \ , \\ c^n_{2_{ij}} &= c_2(A+i\Delta x_1,\ C+j\Delta x_2,\ n\Delta t) \ . \end{aligned}$$

The finite difference equation at a typical grid point then becomes

$$F^{n+1}_{i,j} = F^n_{i+1,j}\left(a^n_{1_{ij}}\frac{\Delta t}{\Delta x_1^2} + b^n_{1_{ij}}\frac{\Delta t}{\Delta x_1}\right) \tag{3-19}$$

$$+ F^n_{i-1,j}\left(a^n_{1_{ij}}\frac{\Delta t}{\Delta x_1^2} - b^n_{1_{ij}}\frac{\Delta t}{\Delta x_1}\right)$$

$$+ F^n_{i,j+1}\left(a^n_{2_{ij}}\frac{\Delta t}{\Delta x_2^2} + b^n_{2_{ij}}\frac{\Delta t}{\Delta x_2}\right)$$

$$+ F^n_{i,j-1}\left(a^n_{2_{ij}}\frac{\Delta t}{\Delta x_2^2} - b^n_{2_{ij}}\frac{\Delta t}{\Delta x_2}\right)$$

$$+ F^n_{i,j}\left(1 - 2a^n_{1_{ij}}\frac{\Delta t}{\Delta x_1^2} - 2a^n_{2_{ij}}\frac{\Delta t}{\Delta x_2^2} + c^n_{ij}\right) + d^n_{ij} \;.$$

The complete grid contains the points (i,j,n) where $n = 0,1,\ldots, N$; $i = 1,2,\ldots, I+1$; and $j = 0,1,2,\ldots, J+1$.

Before solving (3-19) appropriate step sizes need to be selected. The one-dimensional maximum principle, which was introduced in the previous chapter, is easily extended to higher-dimensional equations of the form of (3-15). For general two-dimensional equations of the form of (3-15), the maximum principle states that if

$$a_1(x_1,x_2,t) - \Delta x_1|b_1(x_1,x_2,t)| \geq 0 \;, \tag{3-20}$$

$$a_2(x_1,x_2,t) - \Delta x_2|b_2(x_1,x_2,t)| \geq 0 \;,$$

$$1 - \theta\Delta t\, c(x_1,x_2,t) > 0 \;,$$

and

$$1 - (1-\theta)(2a_1(x_1,x_2,t)\frac{\Delta t}{\Delta x_1^2}$$

$$+ 2a_2(x_1,x_2,t)\frac{\Delta t}{\Delta x_2^2} + \Delta t\, c(x_1,x_2,t)) \geq 0 \;,$$

then the formula is consistent, convergent, and stable. Since an explicit formula, $\theta=0$, is being set up to solve (3-15), the truncation error is

$$\tau = O(\Delta x_1^2) + O(\Delta x_2^2) + O(\Delta t) \ . \tag{3-21}$$

On the plane n (corresponding to $t = n\Delta t$) solution values are stored at only $(1/P)^2$ of the grid points (1/P in each space dimension). This is illustrated in Figure 14 for P=5. Only one fifth of the values necessary on the plane n in order to compute solutions at every $(1/P)^2$ grid points on plane n+1 are stored. The additional required values (represented by the circles in Figure 14) are obtained by fitting polynomials to the stored values (represented by the x's in Figure 14).

Notice that all the information to be generated falls in line with either the vertically or horizontally stored data. This greatly facilitates the polynomial approximation technique. Rather than having to fit a two-dimensional polynomial to n^2 points, it is only necessary to fit two one-dimensional polynomials to n points [42]. That is, instead of generating the entire two-dimensional surface, it is only necessary to produce two lines in the surface.

Figure 15 illustrates how second order polynomials are used to generate the additional required values. The values at (i,j+1,n) and i,j-1,n) are obtained by fitting a second order polynomial to the function values at (i,j+P,n), (i,j-P,n), and (i,j,n). The values at (i+1, j,n) and (i-1,j,n) are obtained in a similar manner. With this information (3-19) can be used to calculate $F_{i,j}^{n+1}$. This procedure can be continued in order to obtain a solution at every $(1/P)^2$ grid point on the plane n+1.

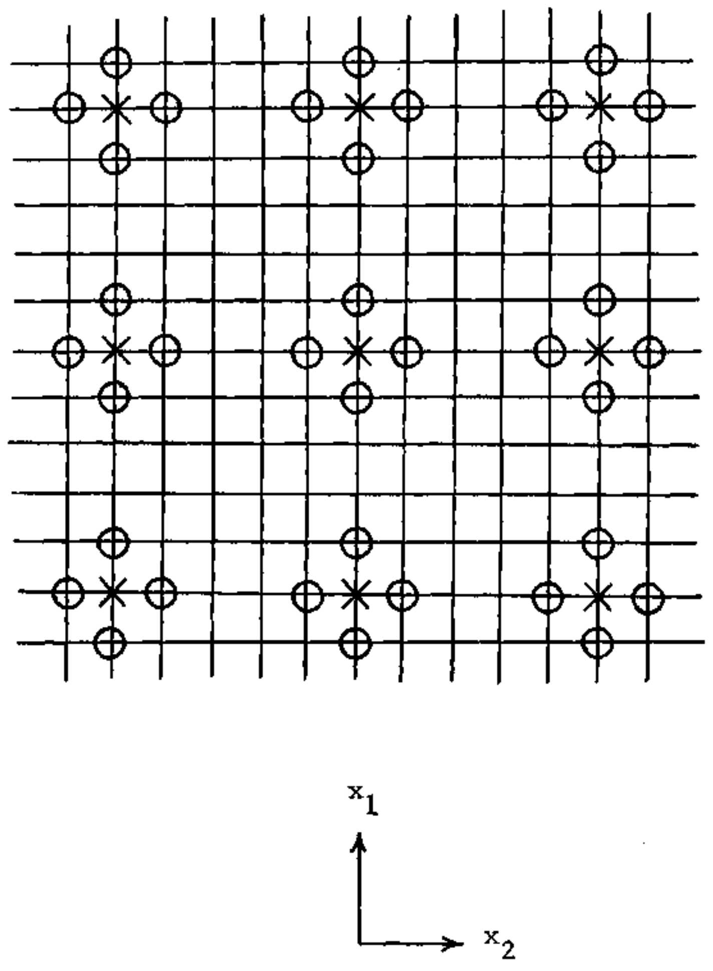


Figure 14. Storage and Regeneration Technique for the Two-Dimensional Algorithm

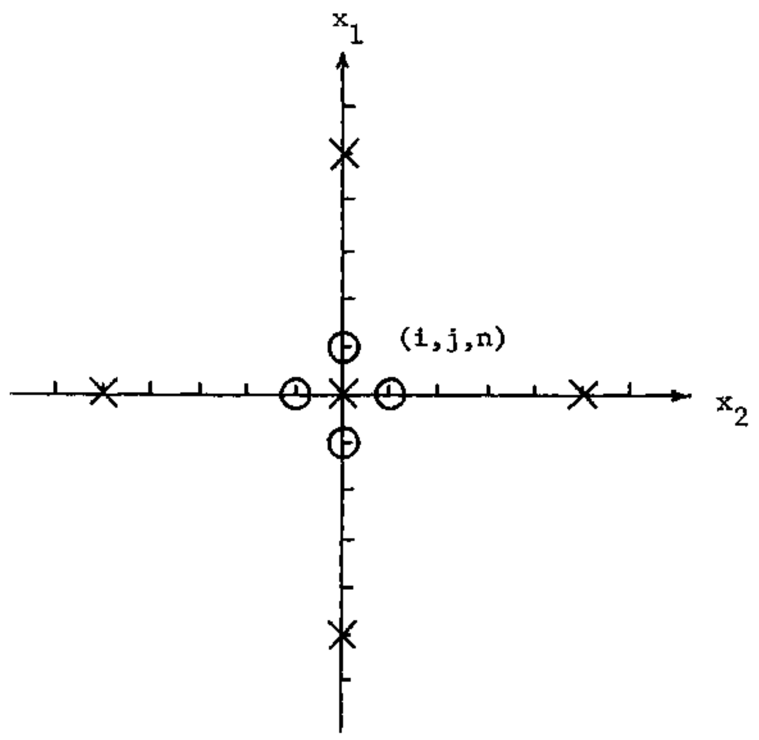


Figure 15. Polynomial Regeneration Technique for the Two-Dimensional Algorithm

Therefore, starting with the initial conditions at n=0, the above method can be used to obtain a complete solution of (3-15).

More Complicated Two-Dimensional Problem

The problem considered in the previous section demonstrates the approach used to solve two-dimensional parabolic partial differential equations. However, as was mentioned earlier, it is desirable when dealing with a two-dimensional Fokker-Planck equation to transform it into an equation containing more complicated terms than are contained in (3-15). In this section an equation is investigated which has all of the type terms which are encountered in Fokker-Planck equations.* The equation considered is

$$F_t(x_1,x_2,t) = a_1(x_1,x_2,t)\, F_{x_1x_1}(x_1,x_2,t) \tag{3-22}$$

$$+ a_2(x_1,x_2,t)\, F_{x_2x_2}(x_1,x_2,t) + b_1(x_1,x_2,t)\, F_{x_1}(x_1,x_2,t)$$

$$+ b_2(x_1,x_2,t)\, F_{x_2}(x_1,x_2,t) + c_1(x_1,x_2,t) \frac{\partial}{\partial x_1}\left(\int_C^{x_2} F(x_1,\varepsilon,t)d\varepsilon\right)$$

$$+ c_2(x_1,x_2,t) \frac{\partial}{\partial x_2}\left(\int_C^{x_1} F(\varepsilon,x_2,t)d\varepsilon\right) ,$$

with initial and boundary conditions

$$F(x_1,x_2,0) = g(x_1,x_2) \ , \qquad A < x_1 < B \quad \text{and} \quad C < x_2 < D \tag{3-23}$$

$$F(A,x_2,t) = h_1(x_2,t) \ , \qquad t > 0 \qquad \text{(continued)}$$

*The additional terms appear if a two-dimensional Fokker-Planck equation is expressed in terms of the probability distribution function rather than the probability density function.

$$F(B,x_2,t) = h_2(x_2,t) \;, \qquad t > 0$$

$$F(x_1,C,t) = h_3(x_1,t) \;, \qquad t > 0$$

$$F(x_1,D,t) = h_4(x_1,t) \;, \qquad t > 0 \;.$$

Notice that the last two terms on the right of (3-22) are of a type not present in (3-15).

Let
$$\int_A^{x_1} F(\varepsilon,x_2,t)d\varepsilon = G(x_1,x_2,t) \;, \tag{3-24}$$

and
$$\int_C^{x_2} F(x_1,\varepsilon,t)d\varepsilon = H(x_1,x_2,t) \;.$$

Using (3-24), (3-23) becomes

$$F_t = a_1\, F_{x_1x_1} + a_2\, F_{x_2x_2} + b_1\, F_{x_1} + b_2\, F_{x_2} + c_1\, H_{x_1} + c_2\, G_{x_2} . \tag{3-25}$$

The normal explicit finite difference equation which approximates (3-25) is

$$F^{n+1}_{i,j} = F^n_{i+1,j}\left(a^n_{1_{ij}} \frac{\Delta t}{\Delta x_1^2} + b^n_{1_{ij}} \frac{\Delta t}{2\Delta x_1}\right) + F^n_{i-1,j}\left(a^n_{1_{ij}} \frac{\Delta t}{\Delta x_1^2} - b^n_{1_{ij}} \frac{\Delta t}{2\Delta x_1}\right) \tag{3-26}$$

$$+ F^n_{i,j-1}\left(a^n_{2_{ij}} \frac{\Delta t}{\Delta x_2^2} + b^n_{2_{ij}} \frac{\Delta t}{2\Delta x_2}\right) + F^n_{i,j-1}\left(a^n_{2_{ij}} \frac{\Delta t}{\Delta x_2^2} - b^n_{2_{ij}} \frac{\Delta t}{2\Delta x_2}\right)$$

$$+ F^n_{i,j}\left(1 - 2a^n_{1_{ij}} \frac{\Delta t}{\Delta x_1^2} - 2a^n_{2_{ij}} \frac{\Delta t}{\Delta x_2^2}\right)$$

(continued)

$$+ H^n_{i+1,j}\left(c^n_{1_{ij}} \frac{\Delta t}{2\Delta x_1}\right) - H^n_{i-1,j}\left(c^n_{1_{ij}} \frac{\Delta t}{2\Delta x_1}\right)$$

$$+ G^n_{i,j+1}\left(c^n_{2_{ij}} \frac{\Delta t}{2\Delta x_2}\right) - G^n_{i,j-1}\left(c^n_{2_{ij}} \frac{\Delta t}{2\Delta x_2}\right) .$$

The numerical solution of (3-26) is approached in the same manner as was done for (3-19). However, for (3-26) there are three functions ($F(x_1,x_2,t)$, $G(x_1,x_2,t)$, and $H(x_1,x_2,t)$) whose values have to be stored. This is in contrast to only one function, $F(x_1,x_2,t)$, for (3-19). As was the case for the standard two-dimensional problem, function values on plane n (for all three: F, G, and H) are stored for only $(1/P)^2$ of the grid points.

The additional information required of each function in order to use (3-26) to compute solutions on plane n+1 is obtained by fitting polynomials to the stored data.

In order to generate $G(x_1,x_2,t)$ and $H(x_1,x_2,t)$ it is necessary to integrate $F(x_1,x_2,t)$ in the appropriate directions. Simpson's rule [25], which provides a tractable integrating algorithm with a high degree of accuracy, is used. Let

$$M(x_i) = M_i \ . \tag{3-27}$$

Simpson's formula is

$$\int_{x_0}^{x_2} M(x)dx = \frac{h}{3}\,(M_0 + 4M_1 + M_2) - \frac{h^5}{90}\,M^{(4)}(\epsilon) , \tag{3-28}$$

where

$$x_{i+1} - x_i = h \ ,$$

$$x_0 < \epsilon < x_2 \ .$$

The first term on the right of (3-28) is the approximation of the integral, and the second term is the error term of the numerical integration. Therefore, the errors associated with the use of Simpson's rule in determining G and H are

$$\frac{h^5}{90} F_{x_1x_1x_1x_1}(\epsilon) = \frac{P^5 \Delta x_1^5}{90} F_{x_1x_1x_1x_1}(\epsilon) \ , \qquad (3\text{-}29)$$

$$x_{1_0} < \epsilon < x_{1_2} \ ,$$

and

$$\frac{h^5}{90} F_{x_2x_2x_2x_2}(\xi) = \frac{P^5 \Delta x_2^5}{90} F_{x_2x_2x_2x_2}(\xi) \ ,$$

$$x_{2_0} < \xi < x_{2_2} \ .$$

CHAPTER IV

LINEAR SYSTEMS

The modified algorithm was tested by solving the Fokker-Planck equation for several linear systems. This is done because the theoretical solution of such systems can be computed very easily when the input is a Gaussian noise [38]. This permits accuracy studies to be made. These studies are also used to determine how much storage should be retained (what value should P be) and what order of polynomial interpolation should be used in order to get the best results with the modified algorithm.

The solution of the Fokker-Planck equation for a first order linear system is studied first. As was done in the previous chapter, two different second order systems are considered. The first second order system has a Fokker-Planck equation which is a normal two-dimensional parabolic partial differential equation. The second, and more complicated, second order system considered has a Fokker-Planck equation of the form of (3-22).

First Order Linear System

The first order linear system considered is shown in Figure 16. Its state equation is

$$\dot{x} = -x + \eta(t) \ . \tag{4-1}$$

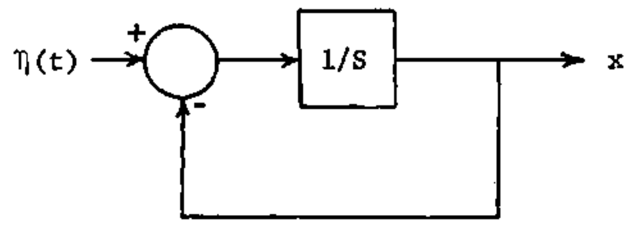


Figure 16. First Order Linear System

The input, $\eta(t)$, is Gaussian white noise with a power spectral height of four. The Fokker-Planck equation which describes the probability density function, f(x,t), of the output, x(t), is

$$\begin{aligned} f_t(x,t) &= (xf(x,t))_x + f_{xx}(x,t) \ , \qquad (4\text{-}2)\\ f(x,0) &= g(x) \ , \\ f(-\infty,t) &= 0 \ , \\ f(\infty,t) &= 0 \ . \end{aligned}$$

The function, g(x), is the initial probability density of the output.

It is more convenient to work with the probability distribution function, F(x,t), rather than with the probability density function, f(x,t). To do this, (4-2) can be changed to an equivalent equation in terms of the distribution function by integrating (4-2) and using the relationship between density and distribution functions. Integrating (4-2)

$$\int_{-\infty}^{x} f_t(\epsilon,t)d\epsilon = \int_{-\infty}^{x} (\epsilon f(\epsilon,t))_\epsilon d\epsilon + \int_{-\infty}^{x} f_{\epsilon\epsilon}(\epsilon,t)d\epsilon \ . \qquad (4\text{-}3)$$

Using the relationship

$$f(x,t) = F_x(x,t) \ , \qquad (4\text{-}4)$$

(4-3) becomes

$$\int_{-\infty}^{x} F_{\epsilon t}(\epsilon,t)d\epsilon = \int_{-\infty}^{x} (\epsilon F_\epsilon(\epsilon,t))_\epsilon d\epsilon + \int_{-\infty}^{x} F_{\epsilon\epsilon\epsilon}(\epsilon,t)d\epsilon \ . \qquad (4\text{-}5)$$

Solving (4-5) yields

$$F_t(\epsilon,t)\Big|_{-\infty}^{x} = F_{xx}(\epsilon,t)\Big|_{-\infty}^{x} + \epsilon F_x(\epsilon,t)\Big|_{-\infty}^{x} . \tag{4-6}$$

Since

$$\lim_{x\to\pm\infty} x^n \frac{\partial^m f(x,t)}{\partial x^m} = \lim_{x\to\pm\infty} x^n \frac{\partial^{m+1} F(x,t)}{\partial x^{m+1}} = 0 , \tag{4-7}$$

$$n \geq 0 \quad \text{and} \quad m \geq 0 ,$$

(4-6) becomes

$$F_t(x,t) = F_{xx}(x,t) + xF_x(x,t) . \tag{4-8}$$

This equation is equivalent to (4-2) except that now the Fokker-Planck equation is in terms of the probability distribution function, F(x,t). The initial and boundary conditions for (4-8) now become

$$F(x,0) = G(x) = \int_{-\infty}^{x} g(\epsilon)d\epsilon , \tag{4-9}$$

$$F(-\infty,t) = 0 ,$$

$$F(\infty,t) = 1 .$$

There are several reasons for preferring to work with the Fokker-Planck equations in terms of distribution functions. Since the distribution function is the integral of the density function, the distribution function is a much smoother curve. Therefore, the numerical calculations are more accurate when the equation with the distribution function, (4-8), is used. Notice that (4-8), the equation using the distribution function, is a somewhat simpler equation than (4-2), the equation using the density function. It is true in general that one-dimensional Fokker-Planck equations using distribution functions are slightly simpler in form than the equivalent equations using density functions. Another important reason for preferring to work with distribution functions is that later when dealing with nonlinear systems the modified formulation makes the boundary conditions easier to understand and apply.

In order to have a specific problem to work with, assume that the initial condition, G(x), is the error function [3] resulting from a Gaussian density function with zero mean and variance σ_0^2. That is

$$F(x,0) = G(x) = \tfrac{1}{2} + \mathrm{Erf}\left(\frac{x}{\sigma_0}\right) = \int_{-\infty}^{x} f(\varepsilon)d\varepsilon \ , \tag{4-10}$$

where

$$f(\varepsilon) = \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{\varepsilon^2}{2\sigma_0^2}} . \tag{4-11}$$

The theoretical solution to this problem, (4-8), with initial and boundary conditions, (4-9) and (4-10), is easily computed [38]. The distribution function of the output, x(t), is

$$F(x,t) = \tfrac{1}{2} + \mathrm{Erf}\left(\frac{x}{\sigma(t)}\right) , \tag{4-12}$$

where

$$\sigma^2(t) = 1 + (\sigma_0^2 - 1)e^{-2t} . \tag{4-13}$$

In order to simulate (4-8) on a computer, the problem must have finite boundaries. Such boundaries are easily selected with the aid of the theoretical results, (4-12). The boundaries must be chosen such that all the probability mass lies within them. Since only Gaussian probabilities are encountered, this is easily done. For example, let the boundaries be at plus and minus four or five standard deviations. Let these boundaries be designated as ±b. Then the problem to be simulated is

$$\begin{aligned} F_t(x,t) &= F_{xx}(x,t) + xF_x(x,t) \ , \qquad (4\text{-}14) \\ F(x,0) &= \tfrac{1}{2} + \mathrm{Erf}\left(\frac{x}{\sigma_0}\right) \ , \\ F(-b,t) &= 0 \ , \\ F(b,t) &= 1 \ . \end{aligned}$$

However, since the probability density function of the output is symmetrical, it is only necessary to solve (4-14) for $-b \le x \le 0$. Therefore, the boundary conditions of (4-14) are replaced by

$$F(-b,t) = 0 \ , \qquad (4\text{-}15)$$

and

$$F(0,t) = 0.5 \ .$$

The Fokker-Planck equation for this first order linear system is used as a test problem for the modified algorithm. The problem is solved using several different amounts of storage (different values for P) and two different orders of polynomial interpolation in an effort to find out which values give the best results. These results of these runs are compared to the theoretical solutions and to solutions obtained by using the original explicit finite difference scheme. For this example, the modified algorithm is studied using interpolating polynomials of order two and three[*] and for storage savings of 80 percent (P=5), 75 percent (P=4), and 66.7 percent (P=3).

The complete results of these computed solutions are tabulated in Appendix A[**]. Tables 1 and 2 summarize the average maximum and mean square errors for the different polynomial interpolations. The errors are averaged over the different values of P (3, 4, and 5). Table 1 gives results from short runs when the initial condition was the steady state solution (summarized from Table 7 in Appendix A). Table 2 lists the results from much longer runs when the initial conditions were such that the solution had to diffuse (inward for half the runs and outward for the other half) to the steady state solution (summarized from Tables 8 and 9 in Appendix A).

Typical solutions from the above runs are illustrated in Figures 17 and 18. Figure 17 shows the results obtained when the modified algorithm was used to solve (4-14) with P=5, $\sigma_0^2 = 0.25$, and using second order polynomial interpolation. Figure 18 illustrates the results obtained when the modified algorithm was used to solve (4-14) with P=5, $\sigma_0^2 = 4$, and using second order polynomial interpolation.

Examining the maximum and mean square errors in Tables 1, 2, 7, 8, and 9, it is seen that in all cases the best results were obtained when second order polynomial interpolation was used. Therefore, in further

---

[*]In Chapter II it was pointed out that the order of interpolating polynomials for equally spaced data should be low.

[**]All computer programs were written in Fortran IV and were executed on a Univac 1108 computer.

Table 1. Average Errors As a Function of $P -- \sigma_0^2 = 1.0$ (from Table 7 in Appendix A)

Degree of Approximating Polynomial	Solution at t=	Average Maximum Error $\times 10^{-3}$	Average Mean Square Error $\times 10^{-10}$
Explicit Finite Difference Scheme	0.025	0.57	167
2	0.025	0.213	206
3	0.025	1.033	2466

Table 2. Average Errors As a Function of $P -- \sigma_0^2 = 0.25$ and 4.0 (from Tables 8 and 9 in Appendix A)

Degree of Approximating Polynomial	Solution at t=	Average Maximum Error $\times 10^{-3}$	Average Mean Square Error $\times 10^{-5}$
Explicit Finite Difference Scheme	0.5	13.06	3.90
2	0.5	9.86	3.01
3	0.5	21.42	15.70

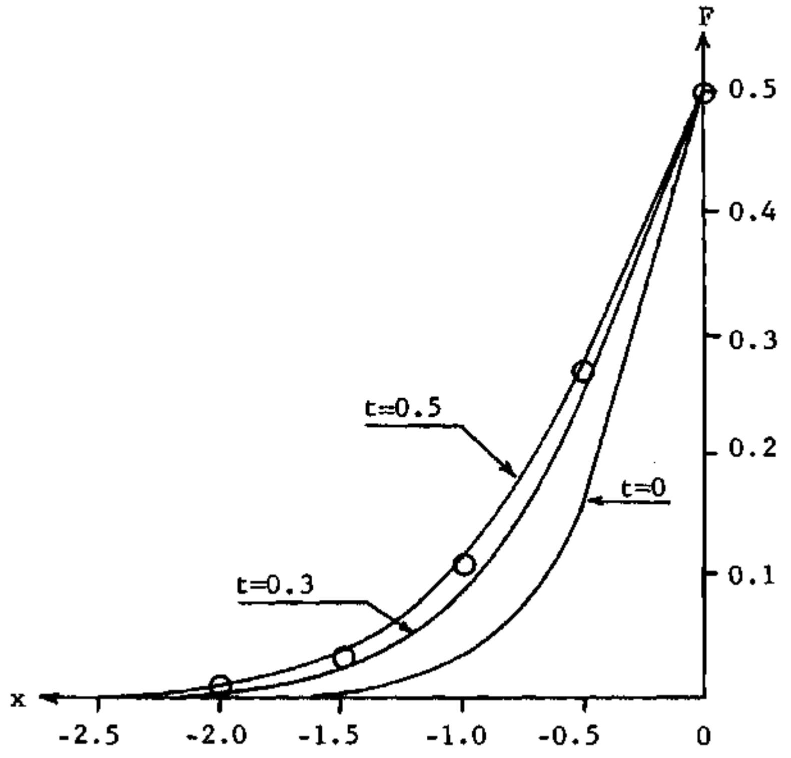


Figure 17. Solution of the Fokker-Planck Equation for the First Order Linear System--$\Delta x = 0.1$, $\Delta t = 0.005$, $\sigma_0^2 = 0.25$ (The solid curves are the computed solutions and the circles represent the true solution at $t = 0.5$.)

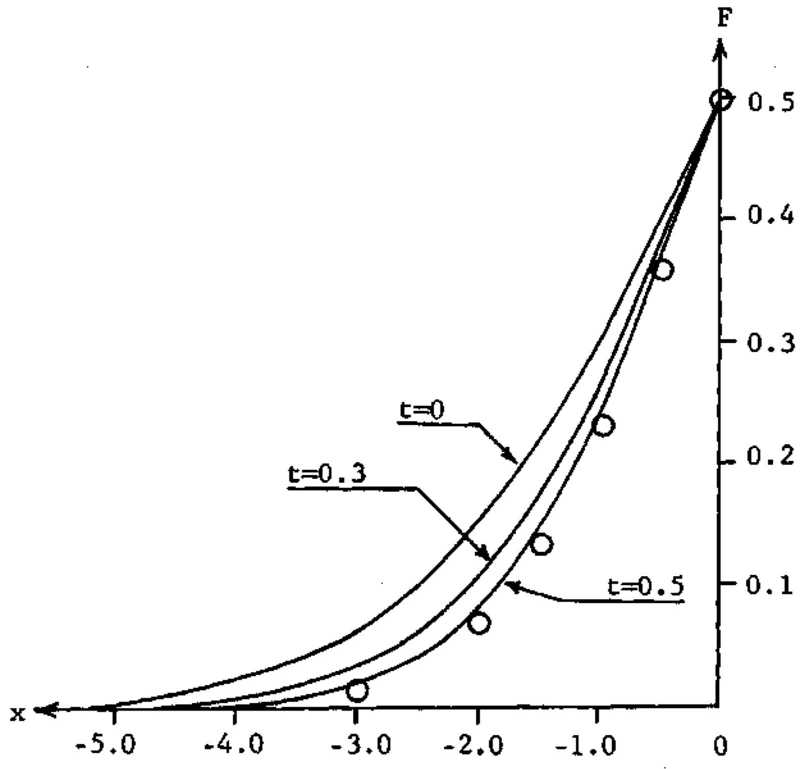


Figure 18. Solution of the Fokker-Planck Equation for the First Order Linear System--$\Delta x = 0.1$, $\Delta t = 0.005$, $\sigma_0^2 = 4.0$ (The solid curves are the computed solutions and the circles represent the true solution at $t = 0.5$.)

calculations only second order polynomial interpolation is used.

The computer runs in this section indicate that the modified algorithm can be used very successfully to solve one-dimensional parabolic partial differential equations. For example, examine in Tables 7, 8, and 9 the results obtained when P was five and when second order polynomial interpolation was used. For these particular cases the modified algorithm gave the same results (comparable accuracy) as the explicit finite difference method while giving savings of 80 percent in computer storage and about 70 percent in computer time.

Simple Second Order Linear System

The first second order linear system considered is a relatively simple system. It is analyzed in order to determine how well the modified formula works for two-dimensional Fokker-Planck equations and to try and determine what percentage of the values at the grid points should be stored (what value P should be). Figure 19 illustrates the system considered. The state equations for this system are

$$\dot{x}_1 = - x_1 + \eta_1(t) , \qquad (4\text{-}16)$$

$$\dot{x}_2 = - x_2 + \eta_2(t) .$$

The inputs, $\eta_1(t)$ and $\eta_2(t)$, are identical, but independent, Gaussian white noises with power special heights of four.

The Fokker-Planck equation representing (4-16) is

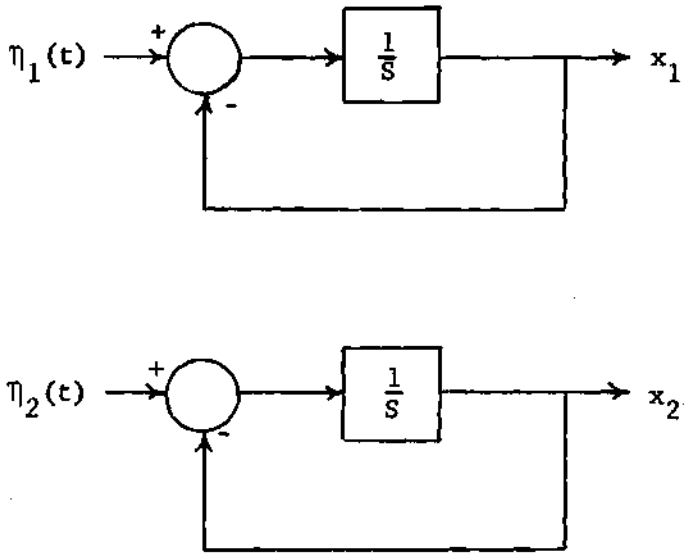


Figure 19. Simple Second Order Linear System

$$f_t(x_1,x_2,t) = (x_1 f(x_1,x_2,t))_{x_1} + f_{x_1x_1}(x_1,x_2,t) \tag{4-17}$$

$$+ (x_2 f(x_1,x_2,t))_{x_2} + f_{x_2x_2}(x_1,x_2,t) \ ,$$

$$f(x_1,x_2,0) = g(x_1,x_2) \ ,$$

$$f(-\infty,x_2,t) = 0 \ ,$$

$$f(x_1,-\infty,t) = 0 \ ,$$

$$f(\infty,x_2,t) = 0 \ ,$$

$$f(x_1,\infty,t) = 0 \ .$$

The function $f(x_1,x_2,t)$ is the joint probability density function of the state of the system. Again it is more convenient to work with the equation in terms of the probability distribution function. Integrating (4-17) with respect to x_1 and x_2 yields

$$F_t(x_1,x_2,t) = x_1F_{x_1}(x_1,x_2,t) + F_{x_1x_1}(x_1,x_2,t) \tag{4-18}$$

$$+ x_2F_{x_2}(x_1,x_2,t) + F_{x_2x_2}(x_1,x_2,t) \ ,$$

$$F(x_1,x_2,0) = G(x_1,x_2) = \int_{-\infty}^{x_2}\int_{-\infty}^{x_1} g(\xi,\varepsilon)d\xi d\varepsilon \ ,$$

$$F(-\infty,x_2,t) = 0 \ ,$$

$$F(x_1,-\infty,t) = 0 \ ,$$

$$F_{x_1}(\infty,x_2,t) = 0 \ ,$$

(continued)

$$F_{x_2}(x_1,\infty,t) = 0 \quad ,$$

$$F(\infty,\infty,t) = 1 \quad ,$$

where $F(x_1,x_2,t)$ is the joint probability distribution function of the state of the system.

The last boundary condition in (4-18) is extraneous. However, it is consistent with the problem being considered. The joint probability density function of the state variables of the system and all of the derivatives of the density function must go to zero as any of the state variables approach infinity. Therefore, it is clear that the right hand side of (4-18) goes to zero as x_1 and x_1 go to infinity. Since the initial value of the distribution function is one as x_1 and x_2 approach infinity, its value, as x_1 and x_2 approach infinity, remains unity for all values of time. This extraneous boundary condition is stated in both this and the next example.

In order to solve (4-18) on a computer, step sizes must be selected and finite boundaries must be established. The maximum principle, (3-23), gives adequate step sizes. Since the theoretical solution for this problem is easily computed, finite boundaries can be set up. Assume that the initial condition is

$$F(x_1,x_2,0) = G(x_1,x_2) = \left(\tfrac{1}{2} + \mathrm{Erf}\left(\frac{x_1}{\sigma_{1_0}}\right)\right)\left(\tfrac{1}{2} + \mathrm{Erf}\left(\frac{x_2}{\sigma_{2_0}}\right)\right) \quad . \tag{4-19}$$

For (4-18) and (4-19) the output distribution function is

$$F(x_1,x_2,t) = \left(\tfrac{1}{2} + \mathrm{Erf}\left(\frac{x_1}{\sigma_1(t)}\right)\right)\left(\tfrac{1}{2} + \mathrm{Erf}\left(\frac{x_2}{\sigma_2(t)}\right)\right) \quad , \tag{4-20}$$

where

$$\sigma_1^2(t) = 1 + (\sigma_{1_0}^2 - 1)e^{-2t} \quad , \qquad \text{(Continued)} \tag{4-21}$$

and

$$\sigma_2^2(t) = 1 + (\sigma_{2_0}^2 - 1)e^{-2t} \quad .$$

Since all the probability mass is contained within a relatively small area, the finite boundaries are easily selected. For example, let the boundaries be at plus and minus 4 or 5 standard deviations for each of the random variables. Let these values be a_1, b_1, a_2, and b_2 for the lower and upper bounds of x_1 and x_2, respectively. The initial and boundary conditions for (4-18) become

$$F(x_1,x_2,0) = G(x_1,x_2) \quad , \tag{4-22}$$

$$F(a_1,x_2,t) = 0 \quad ,$$

$$F(x_1,a_2,t) = 0 \quad ,$$

$$F_{x_1}(b_1,x_2,t) = 0 \quad ,$$

$$F_{x_2}(x_1,b_2,t) = 0 \quad ,$$

$$F(b_1,b_2,t) = 1 \quad .$$

The function $G(x_1,x_2)$ is the same one defined by (4-19).

Since x_1 and x_2 are independent and their density functions are symmetric, the joint probability density function

$$f(x_1,x_2,t) = f_1(x_1,t)\ f_2(x_2,t) \quad , \tag{4-23}$$

is also symmetric. This can be used to reduce the area over which a solution must be obtained. It is only necessary to obtain a solution on

$$a_1 \le x_1 \le 0 \quad , \tag{4-24}$$

$$a_2 \le x_2 \le 0 \quad .$$

The problem simulated on the computer is

$$F_t = x_1 F_{x_1} + F_{x_1 x_1} + x_2 F_{x_2} + F_{x_2 x_2} , \tag{4-25}$$

$$F(x_1,x_2,0) = G(x_1,x_2) ,$$

$$F(a_1,x_2,t) = 0 ,$$

$$F(x_1,a_2,t) = 0 ,$$

$$F(\Delta x_1,x_2,t) - F(0,x_2,t) = F(0,x_2,t) - F(-\Delta x_1,x_2,t) ,$$

$$F(x_1,\Delta x_2,t) - F(x_1,0,t) = F(x_1,0,t) - F(x_1,-\Delta x_2,t) .$$

The results of these computed solutions are tabulated in Appendix A. Table 10 lists the results obtained when the initial condition was the steady state solution. The results of these brief runs are all good, even for the cases when P was very large (15, 20, 25). Table 11 lists the results obtained when the solution had to diffuse outward to steady state. Some qualitative comments about the latter runs are given in Table 3.

Examining Tables 10 and 11, in Appendix A, it is seen that excellent results were obtained when P was equal to five. Since P equal to five gives good results and there are no other set criteria for the selection of P, all further runs are made with P equal to five and using second order polynomial interpolation.

In another interesting computer run the original explicit finite difference scheme was used to try to solve (4-25) with

Table 3. Qualitative Results for Simple Second Order System (from Table 11 in Appendix A)

Amount of Storage P=	Qualitative Results
5	Very good, errors are small.
10	Good, errors are an order of magnitude larger than when P=5.
20	Not so good, the diffusion process is observed but the answer has fairly large errors.

$$\Delta x_1 = \Delta x_2 = 0.25 \ , \tag{4-26}$$

$$\Delta t = 0.0005 \ ,$$

and

$$\sigma_{1_0}^2 = \sigma_{2_0}^2 = 0.25 \ .$$

The values selected for the spatial step sizes do not satisfy the maximum principle, (3-23). The results were that the algorithm was unstable. This is interesting since the modified method effectively used much larger spatial step sizes (Tables 10 and 11) and obtained very good results. This points out what was stated previously, that the stability of the modified algorithm is determined by the step sizes used in the finite difference equation (Δx_i) and not be the spacing between the stored values ($P\Delta x_i$).

More Complicated Second Order Linear System

The problem solved in the previous section, (4-25), demonstrates that the modified algorithm can be used very beneficially to solve two-dimensional Fokker-Planck equations. However, the Fokker-Planck equations encountered later when dealing with second order nonlinear systems are somewhat more complicated than (4-25). Therefore, in this section a linear system which has a Fokker-Planck equation that has every type term that is encountered later is considered.

The system to be considered is illustrated in Figure 20. The input, $\eta(t)$, is Gaussian white noise with a power spectral height of four. The state equations for this system are

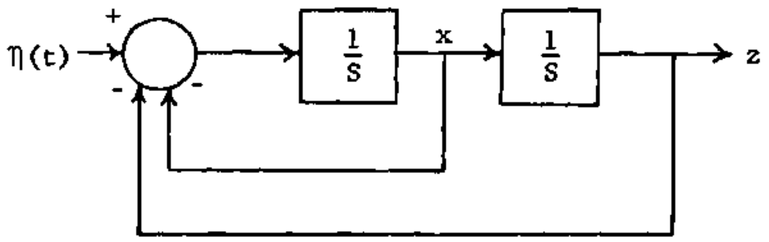


Figure 20. Coupled Second Order Linear System

$$\begin{bmatrix} \dot{z} \\ \dot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \eta(t) . \tag{4-27}$$

The Fokker-Planck equation which describes the joint probability density function of the states of the system is

$$f_t(x,z,t) = - xf_z(x,z,t) + ((x+z)\ f(x,z,t))_x + f_{xx}(x,z,t) , \tag{4-28}$$

$$f(x,z,0) = g(x,z) ,$$

$$f(-\infty,z,t) = 0 ,$$

$$f(x,-\infty,t) = 0 ,$$

$$f(\infty,z,t) = 0 ,$$

$$f(x,\infty,t) = 0 .$$

Upon integration with respect to x and z in order to get it in terms of the distribution function, (4-28) becomes

$$F_t(x,z,t) = (x+z)\ F_x(x,z,t) - xF_z(x,z,t) + F_{xx}(x,z,t) \tag{4-29}$$

$$+ \int_{-\infty}^{x} F_z(\xi,z,t)d\xi - \int_{-\infty}^{z} F_x(x,\epsilon,t)d\epsilon .$$

The two integrals in (4-29) are uniformly convergent; therefore, the order of integration and differentiation can be reversed [44]. Doing this, (4-29) becomes

$$F_t(x,z,t) = (x+z)\ F_x(x,z,t) - xF_z(x,z,t) + F_{xx}(x,z,t) \tag{4-30}$$

$$+ \frac{\partial}{\partial z} \int_{-\infty}^{x} F(\xi,z,t)d\xi - \frac{\partial}{\partial x} \int_{-\infty}^{z} F(x,\epsilon,t)d\epsilon .$$

Let

$$\int_{-\infty}^{x} F(\xi,z,t)d\xi = \overset{x}{F}(x,z,t) \ , \tag{4-31}$$

and

$$\int_{-\infty}^{z} F(x,\epsilon,t)d\epsilon = \overset{z}{F}(x,z,t) \ .$$

When (4-31) is put into (4-30) it simplifies to

$$F_t(x,z,t) = (x+z)\ F_x(x,z,t) - xF_z(x,z,t) \tag{4-32}$$

$$+ F_{xx}(x,z,t) + \overset{x}{F}_z(x,z,t) - \overset{z}{F}_x(x,z,t) \ .$$

The initial and boundary conditions now become

$$F(x,z,0) = G(x,z) = \int_{-\infty}^{z}\int_{-\infty}^{x} g(\xi,\epsilon)d\xi d\epsilon \ , \tag{4-33}$$

$$F(-\infty,z,t) = 0 \ ,$$

$$F(x,-\infty,t) = 0 \ ,$$

$$F_x(\infty,z,t) = 0 \ ,$$

$$F_z(x,\infty,t) = 0 \ ,$$

and

$$F(\infty,\infty,t) = 1 \ .$$

Since the theoretical solution to (4-32) can be computed, finite boundaries can be set up for the problem. Assume that the initial condition for (4-32) is

$$F(x,z,t) = G(x,z) = \left(\tfrac{1}{2} + \text{Erf}\left(\frac{x}{\sigma_{x_0}}\right)\right)\left(\tfrac{1}{2} + \text{Erf}\left(\frac{z}{\sigma_{z_0}}\right)\right) \ . \tag{4-34}$$

The quantities $\sigma_{x_0}^2$ and $\sigma_{z_0}^2$ are the initial variances of the state variables and are to be specified for the particular problem. In accordance with Van Trees [38], the solution to (4-32) is

$$F(x,z,t) = \int_{-\infty}^{z}\int_{-\infty}^{x} f(\xi,\varepsilon,t)d\xi d\varepsilon \ , \tag{4-35}$$

$$f(x,z,t) = \left(\frac{1}{2\pi\sigma_x(t)\ \sigma_z(t)\ \sqrt{1 + r^2(t)}}\right) \times$$

$$\exp\left(-\frac{1}{2(1-r^2(t))}\left(\frac{x^2}{\sigma_x^2(t)} - \frac{2r(t)xz}{\sigma_x(t)\sigma_z(t)} + \frac{z^2}{\sigma_z^2(t)}\right)\right) .$$

For (4-35)

$$\sigma_x^2(t) = 1+K_1e^{-t} + K_2e^{-t}\cos\sqrt{3}t + \frac{K_3-K_2}{\sqrt{3}}e^{-t}\sin\sqrt{3}t \ , \tag{4-36}$$

$$\sigma_z^2(t) = 1+K_4e^{-t} + K_5e^{-t}\cos\sqrt{3}t + \frac{K_6-K_5}{\sqrt{3}}e^{-t}\sin\sqrt{3}t \ ,$$

$$\sigma_{xz}^2(t) = K_7e^{-t} + K_8e^{-t}\cos\sqrt{3}t + \frac{K_9-K_8}{\sqrt{3}}e^{-t}\sin\sqrt{3}t \ ,$$

$$r(t) = \frac{\sigma_{xz}^2(t)}{\sigma_x(t)\sigma_z(t)} \ ,$$

where

$$K_1 = K_4 = \frac{2\sigma_{1_0}^2 + \sigma_{2_0}^2 - 4}{3} \ , \tag{4-37}$$

$$K_2 = \frac{\sigma_{1_0}^2 - 2\sigma_{2_0}^2 + 1}{3} \ ,$$

(continued)

$$K_3 = \frac{4 - 2\sigma^2_{1_0} - 2\sigma^2_{2_0}}{3} ,$$

$$K_5 = \frac{\sigma^2_{2_0} - 2\sigma^2_{1_0} + 1}{3} ,$$

$$K_6 = \frac{- \sigma^2_{1_0} - \sigma^2_{2_0} + 2}{3} ,$$

$$K_8 = - K_7 = \frac{\sigma^2_{1_0} + \sigma^2_{2_0} - 2}{3} ,$$

$$K_9 = \frac{4\sigma^2_{1_0} - 2\sigma^2_{2_0} - 2}{3} .$$

With the aid of (4-35), (4-36), and (4-37) finite boundaries for the problem can be established. For example, let the boundaries be at plus and minus four or five standard deviations of each state variable. Let these lower and upper bounds for x and z be a_1, b_1, a_2, and b_2, respectively. Equation (4-33) now becomes

$$F(x,z,0) = G(x,t) , \tag{4-38}$$

$$F(a_1,z,t) = 0 ,$$

$$F(x,a_2,t) = 0 ,$$

$$F_x(b_1,z,t) = 0 ,$$

$$F_z(x,b_2,t) = 0 ,$$

and

$$F(b_1,b_2,t) = 1 .$$

The problem to be simulated on the computer is (4-32) with initial and boundary conditions (4-38). The function $G(x,z)$ is that which is defined by (4-34).

In order to solve (4-32) numerically, adequate step sizes in the space variables and time must be selected. The problem is no longer of the form so that the maximum principle, (3-23), can be used to assure an adequate selection. However, (3-23) can be used to obtain a good guess for the step sizes. The step size Δx is obtained from (3-23) by letting the coefficient of F_{xx} be $a_1(x_1,x_2,t)$ and the coefficient of F_x be $2b_1(x_1,x_2,t)$. The other step sizes are obtained by letting

$$\Delta z = \Delta x \ , \qquad (4\text{-}39)$$

$$\Delta t \le \frac{\Delta x^2}{4a_1(x_1,x_2,t)} \ .$$

The initial selection for step sizes is

$$\Delta x = \Delta z = 0.04 \ , \qquad (4\text{-}40)$$

and
$$\Delta t = 0.0004 \ .$$

Using the values (4-40), (4-32) was solved when

$$\sigma^2_{x_0} = \sigma^2_{z_0} = 0.25 \ . \qquad (4\text{-}41)$$

The initial conditions are such that the solution had to diffuse outward to steady state. The results were very good (tabulated in Table of 12 of Appendix A).

Since (4-32) lacks a second partial derivative with respect to z, the development of the maximum principle suggests that the algorithm should be stable for Δt twice as large as that given by (4-39). Therefore, (4-32) with conditions (4-41) was solved with

$$\Delta x = \Delta z = 0.04 \,, \qquad (4\text{-}42)$$
$$\Delta t = 0.0008 \,.$$

The results were as good as those obtained using the smaller time step size, (4-40).

Solutions were run using the same space step sizes and successively larger time step sizes until the algorithm became unstable. Some very surprising results were obtained. The algorithm remained stable and accurate until

$$\Delta t = \Delta x = \Delta z \,. \qquad (4\text{-}43)$$

This is rather surprising considering the ratio of the step size selections which is generally recommended by the maximum principle. However, two points should be remembered. First, the equation being considered, (4-32), is not of the exact form for which the maximum principle holds. Also, the maximum principle ensures all the desirable properties if the step sizes are selected in accordance with its rules. If the step sizes are selected by some other criterion, the maximum principle yields no information, either good or bad, about the algorithm.

A summary of the runs discussed above is given in Table 12 (in Appendix A). Note that all computer runs used the modified algorithm with P equal to five (96 percent savings in storage) and with second order polynomial interpolation. Simpson's rule was used to perform the numerical integrations in order to calculate $\overset{X}{F}$ and $\overset{Z}{F}$.

The error columns in Table 12 indicate that the calculated solutions are very good. Actually, the true errors may be slightly better or worse than is shown in Table 12. This is because the true solution was tabulated to only four decimal places and the numerical calculations were carried out to five decimal places. Notice that the errors in the calculated solution did not increase as Δt was increased from 0.0004 to 0.002. This would indicate that the error in the numerical algorithm might have little dependence on the time step size. Several computer runs were made to investigate this possibility further. The results of these runs are tabulated in Tables 13 and 14, in Appendix A.

Notice that in both Tables 13 and 14 solutions are presented for more than one time (in Table 13 solutions are presented for $t = 0.2$ and $t = 1.0$). The accuracy of the solutions at different times should not be compared. Only information about solutions at the same time should be compared.

Tables 12, 13, and 14 illustrate conclusively that the time step size, Δt, for equations of the form (4-45) can be many times larger than that suggested by the maximum principle. These tables also suggest that the errors in the numerical calculations are not a strong function of the choice of Δt (so long as the algorithm remains stable). As a matter of fact, many of the solutions using the smaller step sizes in time are less accurate than those using larger time steps. For those cases the calculations using the smaller step sizes are accumulating larger roundoff errors due to more calculations. However, it was observed that in runs which used values of Δt that were close to the maximum value for stability, errors became much larger for long computer runs (Table 14 in Appendix A). This indicates that the selection of Δt should be somewhat smaller (by a factor of three) than the maximum possible value for stability. Some of the results listed in Tables 13 and 14 are summarized in Table 4.

Tables 15 and 16 list the results of runs which investigated the accuracy of the numerical method as the space step sizes were varied. Examining the errors in these tables it is obvious that changing the space step sizes does drastically affect the accuracy of the numerical results. As the spatial step sizes are increased the error in the solution is also increased.

Typical solutions for the marginal probability distribution functions are shown in Figures 21 and 22. These marginal distributions are easily obtained from the joint distribution function since

$$F_1(x,t) = F(x,b_2,t) \ , \qquad (4\text{-}44)$$

$$F_2(z,t) = F(b_1,z,t) \ .$$

Summary of Results for Linear Systems

The modified algorithm produced very good results when used to solve one- and two-dimensional Fokker-Planck equations. The best results were obtained when a second order interpolating polynomial was used. It was also decided to set P equal to five. These values gave as good results for the one-dimensional and simple two-dimensional problems as did the explicit finite difference algorithm.

Table 4. A Study of Accuracy As a Function of Δt (from Tables 13 and 14 in Appendix A)

Space Step Sizes $\Delta x = \Delta z =$	Time Step Size $\Delta t =$	Solution at t=	Maximum Error $\times 10^{-3}$	Mean Square Error $\times 10^{-10}$
0.04	0.0008	0.2	1.29	865.65
0.04	0.01	0.2	0.67	218.4
0.04	0.01	1.0	2.32	6631.6
0.04	0.02	1.0	2.4	6127.1
0.04	0.04	1.0	U N S T	A B L E
0.1	0.002	1.0	7.52	85300
0.1	0.02	1.0	7.68	83800
0.1	0.05	1.0	8.07	81500

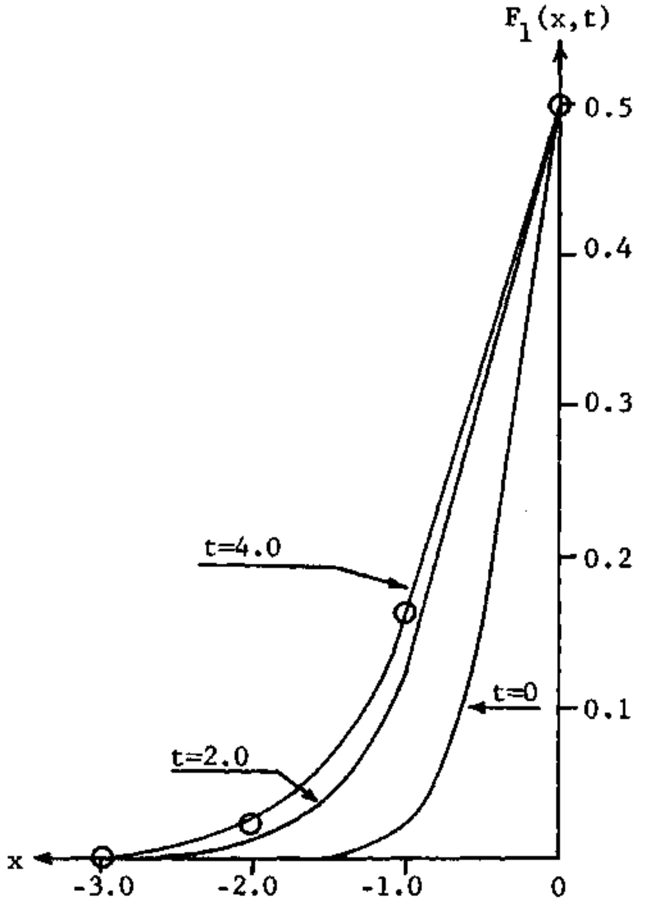


Figure 21. Marginal Probability Distribution for the Coupled Second Order Linear System--$F_1(x,t)$, $\Delta x = \Delta z = 0.1$, $\Delta t = 0.02$, $\sigma^2_{x_0} = \sigma^2_{z_0} = 0.25$

(The solid curves are the computed solutions and the circles represent the true solution at $t = 4.0$.)

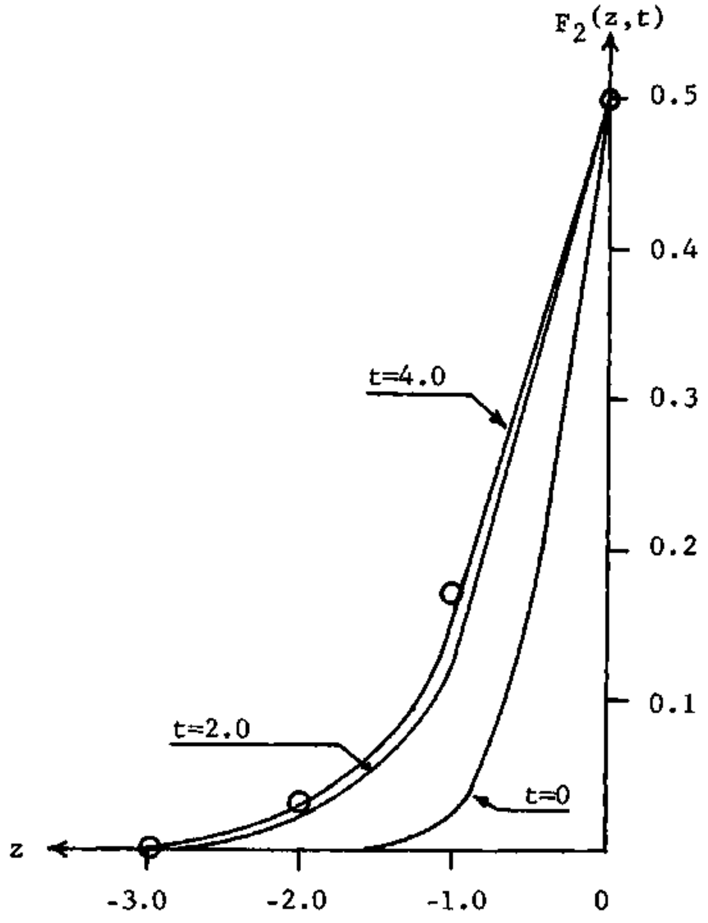


Figure 22. Marginal Probability Distribution for the Coupled Second Order Linear System--$F_2(z,t)$, $\Delta x = \Delta z = 0.1$, $\Delta t = 0.02$, $\sigma^2_{x_0} = \sigma^2_{z_0} = 0.25$

(The solid curves are the computed solutions and the circles represent the true solution at t = 4.0.)

The solution to the more complicated two-dimensional Fokker-Planck equation produced some very surprising and helpful results. It was found that the time step size, Δt, could be much larger than that suggested by the maximum principle. The algorithm remained stable so long as the time step was smaller than the spatial step sizes. It was also observed that the errors in the numerical calculations were not very dependent on Δt (so long as the algorithm remained stable). The accuracy of the algorithm was practically a function of the spatial step sizes.

CHAPTER V

NONLINEAR SYSTEMS

A practical and important application of the Fokker-Planck equation is in the analysis of phased-locked loops in the presence of noise. The solution of the Fokker-Planck equation for a phase-locked loop gives the time varying probability density function of the phase error of the system.

In this chapter the modified algorithm is first used to solve the Fokker-Planck equations for a first and second order phase-locked loop. The probability density and distribution functions of the phase errors are obtained and plotted as functions of time for several signal to noise ratios. The variances of the phase errors are also calculated and plotted as functions of time.

The modified algorithm is then used to solve the Fokker-Planck equations for a first and second order gated phase-locked loop. These results simulate the statistics of the phase errors for phase-locked loops operating in Time Division Multiple Access systems. The variances of the phase errors are plotted as functions of time for several signal to noise ratios.

First Order Phase-Locked Loop

The first order phase-locked loop and its equivalent block diagram are illustrated in Figures 5 and 6 respectively. The input noise, $\eta(t)$, is Gaussian white noise with a power spectral height of N_o. The

Fokker-Planck equation which describes the probability density function of the phase error of this system (given by (1-14) and (1-15)) is

$$f_t(\phi,t) = -\left(\left(\theta_{1_t}(t) - AK\ \sin(\phi)\right) f(\phi,t)\right)_\phi \tag{5-1}$$

$$+ \frac{N_o K^2}{4}\ f_{\phi\phi}(\phi,t)\ .$$

The variables in and related to (5-1) are defined by (1-8).

It is assumed that the received signal is a constant sinusoid of known frequency. That is

$$\theta(t) = \omega t + \theta_o, \tag{5-2}$$

$$\theta_1(t) = (\omega - \omega_o)\ t + \theta_o,$$

and

$$\omega_o = \omega\ .$$

Using (5-2) and making the change of variables

$$\tau = AKt, \tag{5-3}$$

(5-1) becomes

$$f_\tau(\phi,\tau) = (\sin(\phi)\ f(\phi,\tau))_\phi + \frac{KN_o}{4A}\ f_{\phi\phi}(\phi,\tau). \tag{5-4}$$

The term

$$\frac{4A}{N_o K} = \alpha \tag{5-5}$$

is the signal to noise ratio in the first order loop [8]. Putting (5-5) into (5-4) gives

$$f_\tau(\phi,\tau) = (\sin(\phi)\ f(\phi,\tau))_\phi + \frac{1}{\alpha} f_{\phi\phi}(\phi,\tau) \ . \tag{5-6}$$

This is the form in which the Fokker-Planck equations for first order phase-locked loops is generally presented.

The statistics of the phase error, ϕ, are sought on modulo 2π [8] [41]. This means that the phase error is always interpreted as being between $-\pi$ and $+\pi$. The process that this represents is the phase error which would be indicated by a phase meter (illustrated in Figures 23 and 24). This is a very natural region on which to seek a solution since at any given instant the best that can be hoped for is to match the phase of the reference signal to the phase of the current cycle of the received signal.

Before a solution to (5-6) can be found, adequate initial and boundary conditions have to be specified. These have to be established from physical considerations of the system. A logical set of conditions which is generally used, [8] [41] [12], is

$$f(\phi,0) = g\ (\phi) \quad , \qquad -\pi \le \phi \le \pi \tag{5-7}$$

$$f(-\pi,\tau) = f(\pi,\tau) \ ,$$

and

$$\int_{-\pi}^{\pi} f(\phi,\tau)d\phi = 1 \quad .$$

The function $g(\phi)$ is an initial density function which has to be specified. No apriori knowledge is assumed for the initial phase error.

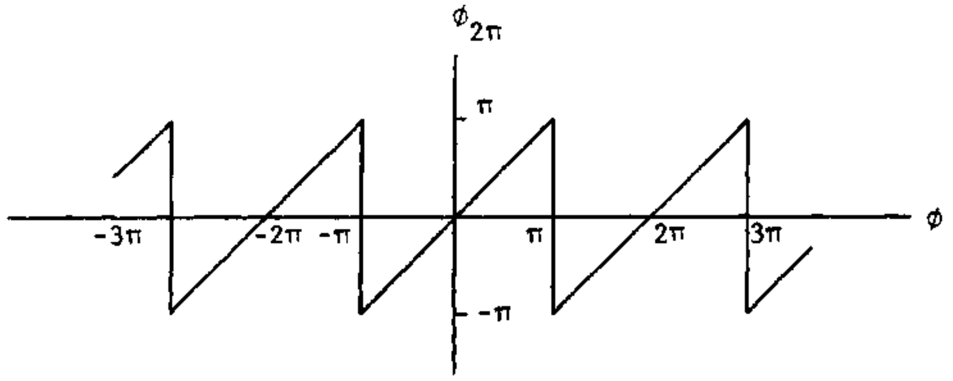


Figure 23. Relationship Between the Phase Error and the Modulo 2π Phase Error

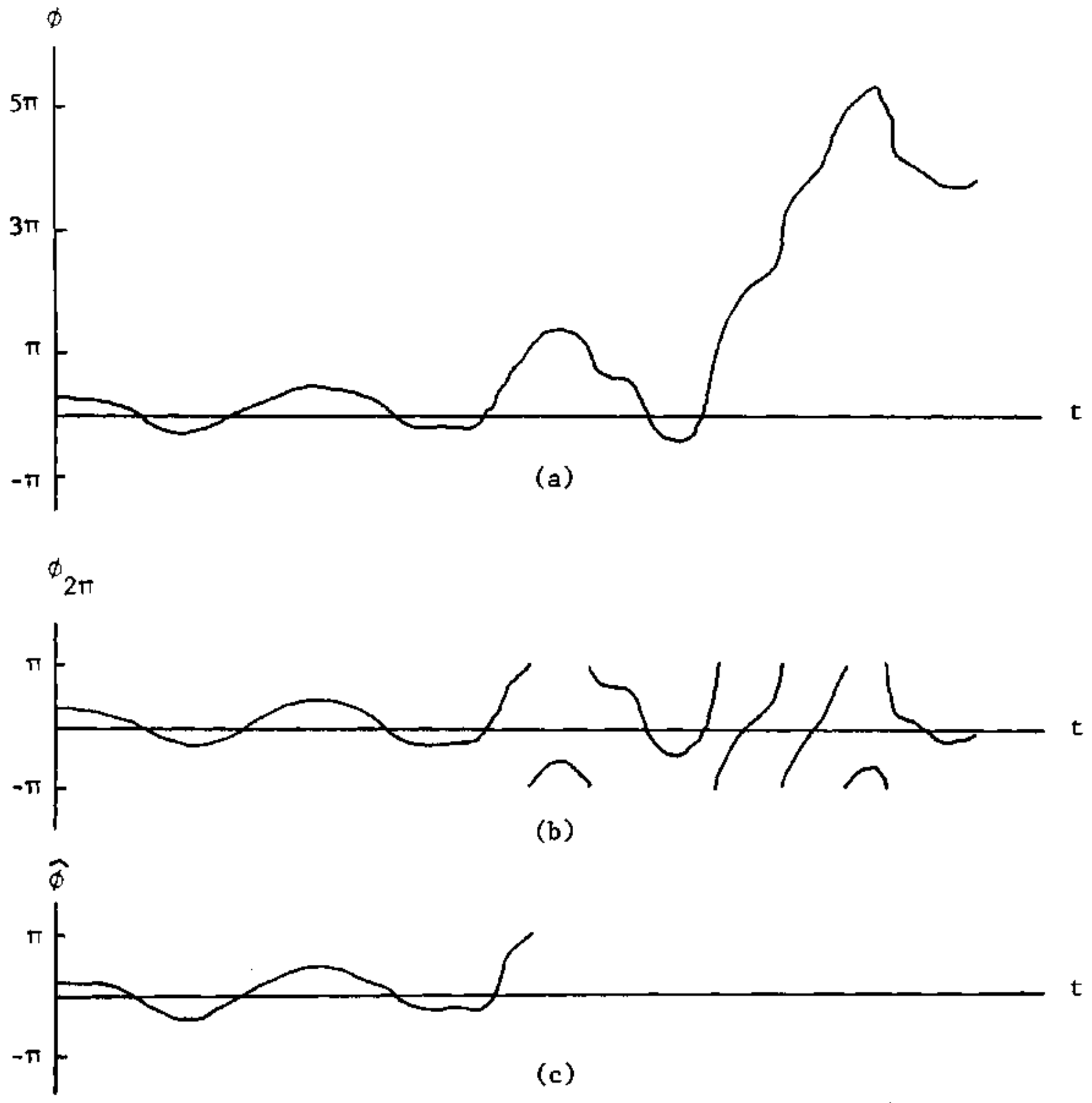


Figure 24. a) One Possible Phase Trajectory
b) Modulo 2π Phase Trajectory
c) Phase Trajectory for a System with an Absorbing Boundary

Therefore, $g(\phi)$ is uniformly distributed. That is

$$g(\phi) = 1/2\pi \; , \quad -\pi \le \phi \le \pi \quad . \tag{5-8}$$

In order to solve for the statistics of the phase error it is more convenient to change (5-6) and (5-7) into equivalent equations in terms of probability distribution functions. This is accomplished by integrating (5-6) with respect to ϕ (from $-\pi$ to an arbitrary point $\phi \le \pi$). The result is

$$F_\tau(\phi,\tau) = \sin(\phi)\, F_\phi(\phi,\tau) + \frac{1}{\alpha} F_{\phi\phi}(\phi,\tau) - J(-\pi,\tau) \quad , \tag{5-9}$$

where $F(\phi,\tau)$ is the probability distribution function of the phase error.

The function $J(\phi,\tau)$ is the "probability current" [41] at the point ϕ. The probability current is the amount of probability per unit time passing through the point ϕ. For this particular problem, the probability current is given by

$$J(\phi,\tau) = \sin(\phi)\, f(\phi,\tau) + \frac{1}{\alpha} f_\phi(\phi,\tau) \quad . \tag{5-10}$$

Due to the symmetry of the problem being considered, it is clear that the probability current on the boundaries is zero. Lindsey [41] arrives at the same conclusion mathematically for the steady state case.

Therefore, the equation and boundary conditions, in terms of the distribution function, which are equivalent to (5-6) and (5-7) are

$$F_\tau(\phi,\tau) = \sin(\phi)\; F_\phi(\phi,\tau) + \frac{1}{\alpha} F_{\phi\phi}(\phi,\tau) \quad , \qquad (5\text{-}11)$$

$$F(\phi,0) = (\phi + \pi)/2\pi \quad , \qquad -\pi \le \phi \le \pi$$

$$F(-\pi,\tau) = 0 \quad ,$$

and
$$F(\pi,\tau) = 1 \quad .$$

The modified algorithm* was used to obtain a solution to (5-11).** The step sizes ($\Delta\phi$ and $\Delta\tau$) used in the numerical calculations were selected so as to satisfy the maximum principle, (2-15). Due to the symmetry of this problem it is only necessary to obtain a solution for $-\pi \le \phi \le 0$.

The steady state solutions to Fokker-Planck equations for first order phase-locked loops are known [8] [41]. The theoretical steady state solution of (5-6) subject to (5-7) and (5-8) is [8]

$$f(\phi) = \frac{\exp(\alpha \cos(\phi))}{2\pi\, I_o(\alpha)} \quad , \qquad (5\text{-}12)$$

where $I_o(\alpha)$ is the zeroth order modified Bessel function of the signal to noise ratio.

---

*The details of the numerical solutions obtained in this chapter are tabulated in Appendix B.

**The computer programs used to obtain the results in this chapter are shown in Appendix C.

Figures 25, 26, and 27 show the results obtained when the modified algorithm was used to solve (5-10) for signal to noise ratios of 0.5, 1.0, and 2.0 respectively. The theoretical steady state results are also indicated on the figures. The computed and theoretical steady state solutions agree very closely. The maximum and mean square errors for the computed steady state results are listed in Table 5.

Pickholtz and Dominiak [12] obtained transient solutions (numerically) to the Fokker-Planck equation for the first order phase-locked loop (they solved (5-6) subject to (5-7)). The modified algorithm was used to obtain a solution to (5-10), with $\alpha = 1$, which could be differentiated and compared to one of their transient solutions. The two solutions, which agree very closely, are shown in Figure 28.

The solutions to the Fokker-Planck equations for the phase-locked loops are used to obtain measures which evaluate the performances of the systems. The most desirable results would be information about the frequency of skipping cycles for a given density function. Such results would illustrate clearly how the phase-locked loop responds as a function of time. While the procedure for the solution of this problem has been developed [8] [45], the actual numerical solution is quite a formidable task. It requires the solution of a two-point boundary value problem for each time which the frequency of skipping cycles is desired. The same problem becomes considerably more difficult for second order phase-locked loops. For this case a one-dimensional partial differential equation has to be solved for each point where the frequency of skipping cycles is desired.

While not being as desirable as the frequency of skipping cycles,

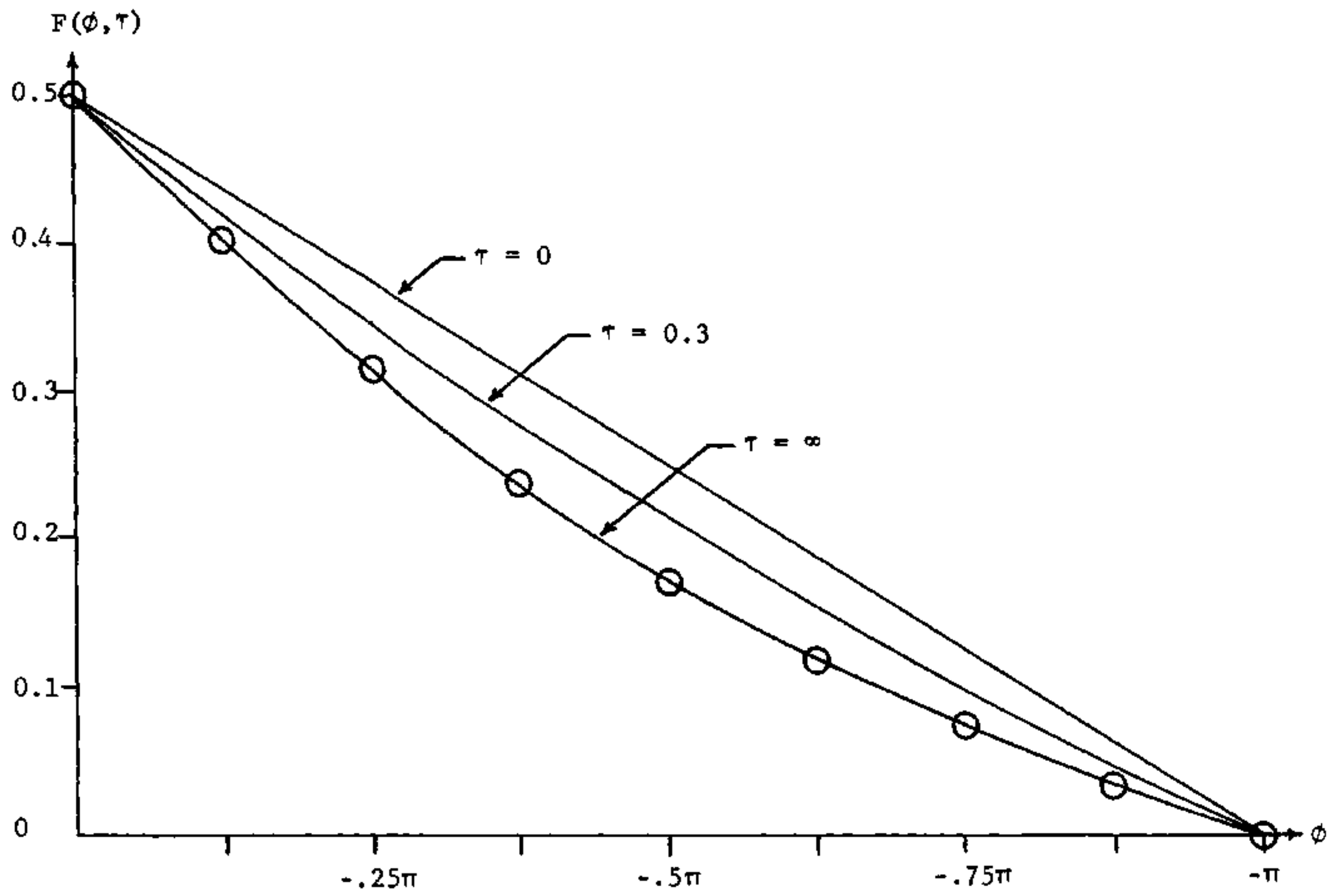


Figure 25. Solution of the Fokker-Planck Equation for the First Order Phase-Locked Loop--$\Delta\phi = \pi/40$, $\Delta\tau = 0.0015$, $\alpha = 0.5$. (The circles are the theoretical steady state solution.)

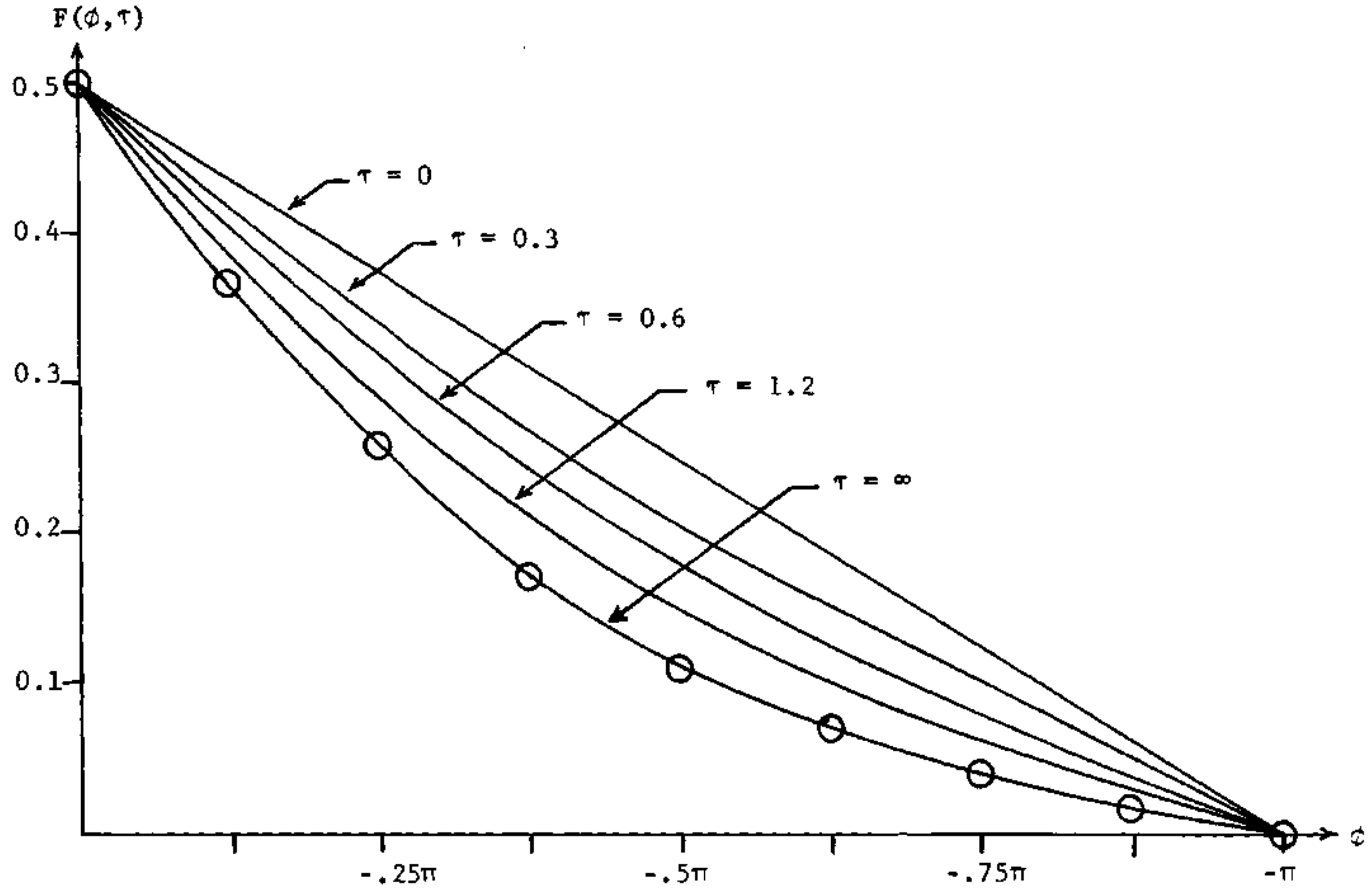


Figure 26. Solution of the Fokker-Planck Equation for the First Order Phase-Locked Loop--$\Delta\phi = \pi/40$, $\Delta\tau = 0.003$, $\alpha = 1.0$. (The circles are the theoretical steady state solution.)

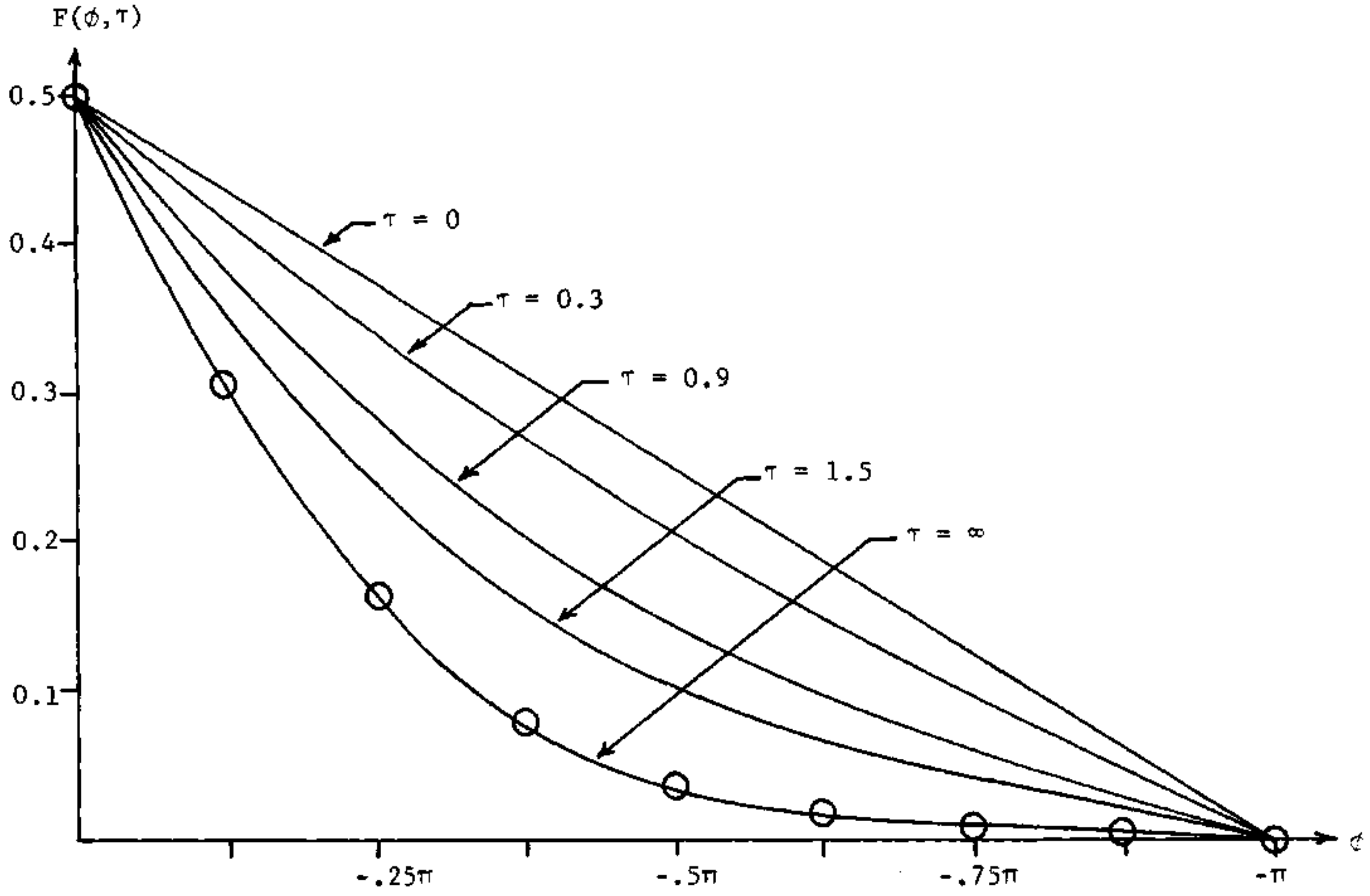


Figure 27. Solution of the Fokker-Planck Equation for the First Order Phase-Locked Loop--$\Delta\phi = \pi/40$, $\Delta\tau = 0.003$, $\alpha = 2.0$. (The circles are the theoretical steady state solution.)

Table 5. Results of the Steady State Solution for a First Order Phase-Locked Loop

Signal to Noise Ratio $\alpha =$	Maximum Error $\times 10^{-3}$	Mean Square Error $\times 10^{-5}$
0.5	1.25	0.065
1.0	3.2	0.392
2.0	8.83	2.234

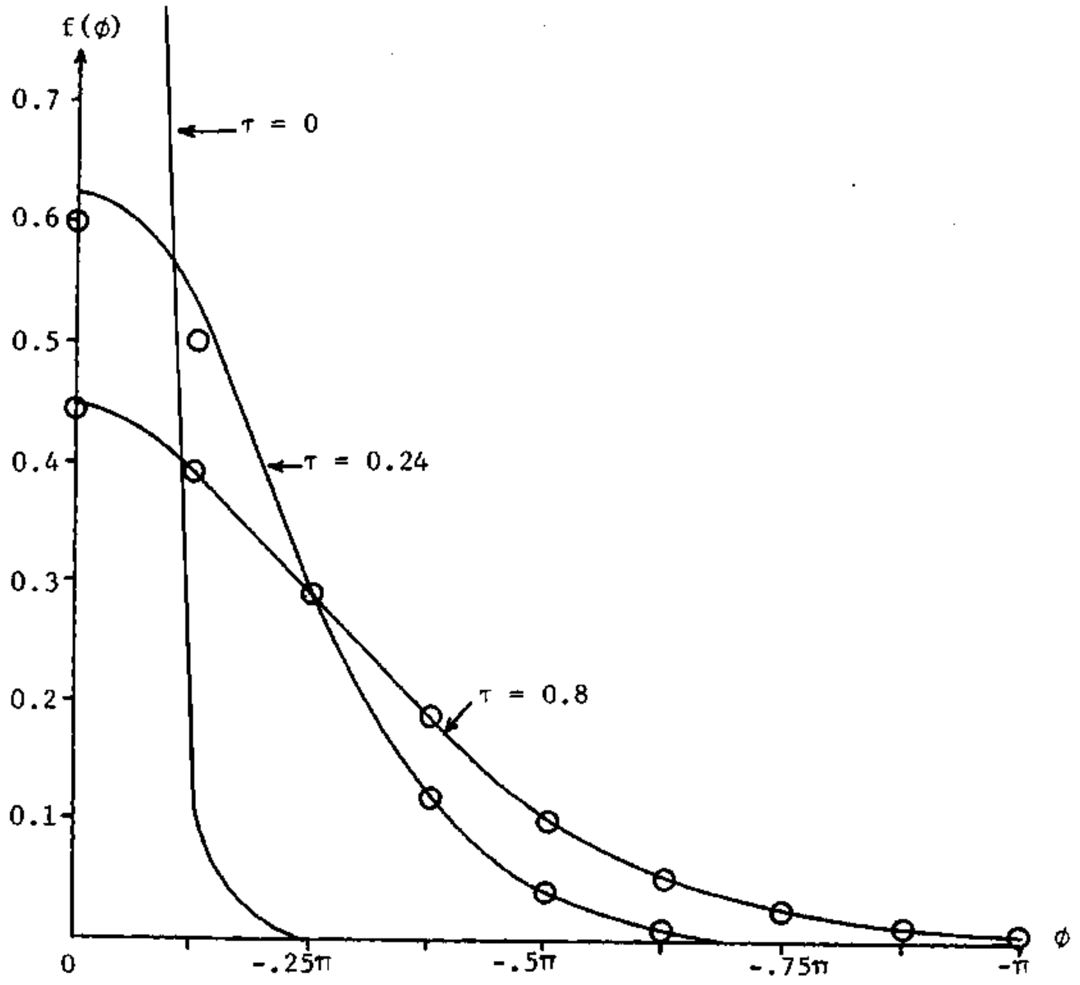


Figure 28. Comparison of a Transient Solution to the Fokker-Planck Equation for the First-Order Phase-Locked Loop With a Transient Solution that Appears in the Literature--$\Delta\phi = \pi/40$, $\Delta\tau = 0.002$, $\alpha = 1$. (The solid curves are Pickholtz and Dominiak's solution, and the circles are the values obtained with the modified algorithm.)

the quantity commonly used to gauge the performance of a phase-locked loop is the variance of the phase error. Figure 29 shows the variances as functions of time for the processes that were illustrated in Figures 25, 26, and 27. The steady state values of these variances agree very closely with the theoretical results displayed by Viterbi [8].

Second Order Phase-Locked Loop

The Fokker-Planck equation for second order phase-locked loops is a very formidable problem, for which to date, no exact theoretical solutions or no complete numerical solutions have been presented.* In this section the modified algorithm is used to obtain complete solutions to the Fokker-Planck equation for a second order phase-locked loop. Results are obtained for several different signal to noise ratios.

The second order phase-locked loop is illustrated in Figure 30 where the linear filter is first order. The transfer function of the linear filter for the loop being considered is

$$\mathcal{L}(h(t)) = 1 + \frac{a}{s} \quad . \tag{5-13}$$

The equivalent block diagram for this system is illustrated in Figure 30. Again the noise input is Gaussian white noise with a power spectral height of N_o.

The dynamic response of this system is described by (1-7) with the appropriate impulse response for the linear filter. It is again

*Lindsey and Charles [26] have presented some experimental density and distribution functions for the steady state phase error of a second order phase-locked loop.

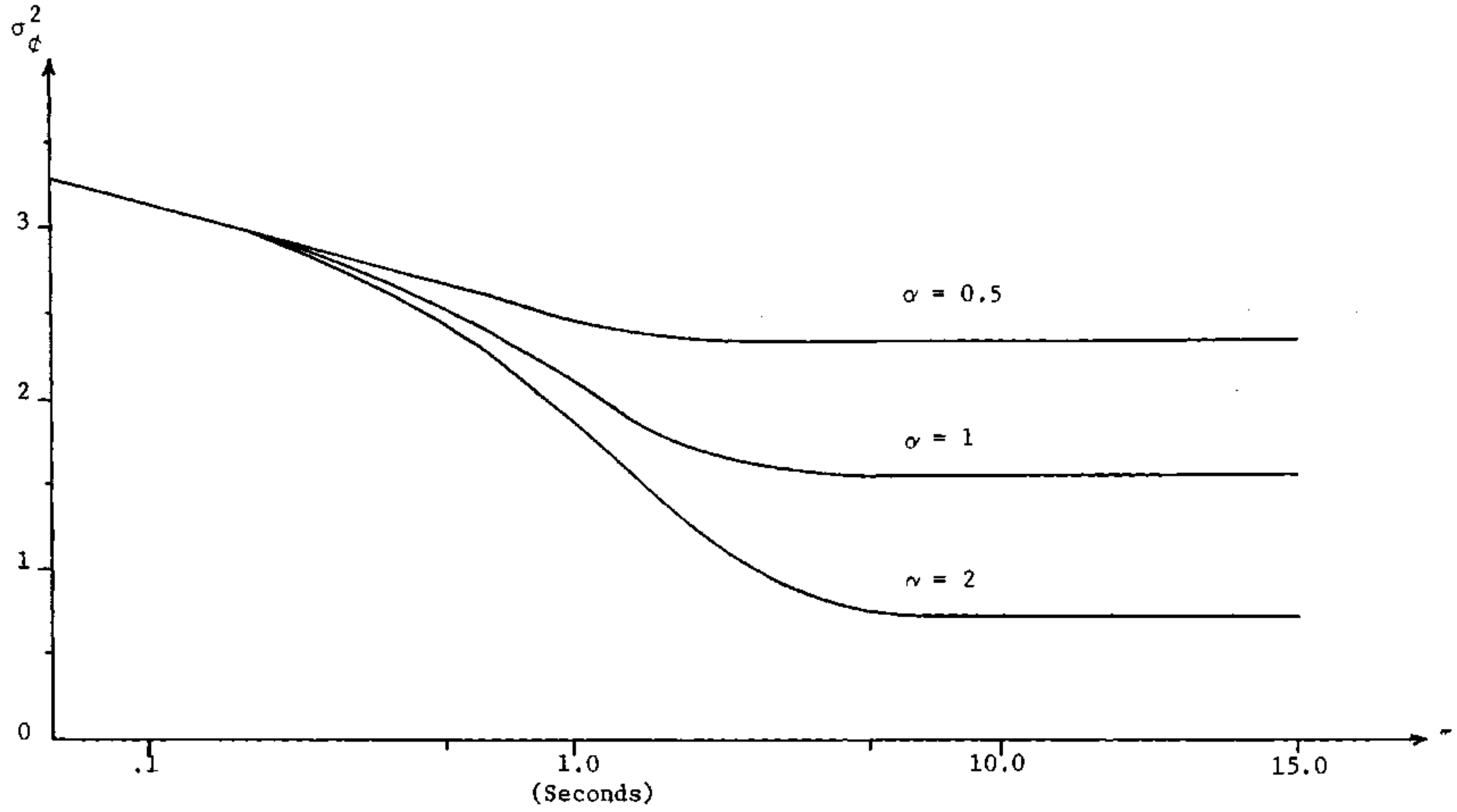


Figure 29. Variances of the Phase Errors for the First Order Phase-Locked Loop

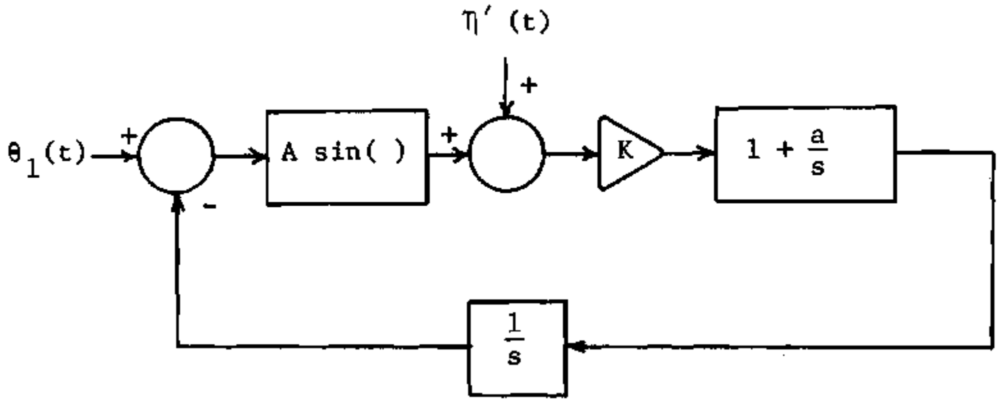


Figure 30. Equivalent Block Diagram of the Perfect Second Order Phase-Locked Loop

assumed that the received signal is a constant sinusoid of known frequency. Therefore,

$$\dot{\theta}_1(t) = 0 \quad . \tag{5-14}$$

Using (5-14) and the impulse response of the loop filter, the equation which describes the operation of the system becomes

$$\dot{\phi}(t) = -K(A\ \sin(\phi(t)) + \eta'(t)) - aAK\int_0^t (\sin(\phi(u)) + \eta'(u))du \quad . \tag{5-15}$$

Equation (5-15) is put in a more tractable form by using a change of variables described by Viterbi [8]. Defining

$$\phi = \dot{\epsilon}(t) + a\ \epsilon(t) \quad , \tag{5-16}$$

(5-15) becomes

$$\ddot{\epsilon}(t) + a\dot{\epsilon}(t) = -K(A\ \sin(\dot{\epsilon}(t) + a\,\epsilon(t)) + \eta'(t)) - aK\int_0^t (A\ \sin(\dot{\epsilon}(u) + a\epsilon(u)) + \eta'(u))du \ . \tag{5-17}$$

This equation can be separated into two equations which, to within an arbitrary constant, have the same solution. These are

$$\ddot{\epsilon}(t) = -K(A\sin(\dot{\epsilon}(t) + a\epsilon(t)) + \eta'(t)) \quad , \qquad (5\text{-}18)$$

and

$$a\dot{\epsilon}(t) = -aK\int_o^t A\sin(\dot{\epsilon}(u) + a\epsilon(u)) + \eta'(u)du \quad .$$

Defining the variables

$$y_o = \epsilon(t) \qquad (5\text{-}19)$$

and

$$y_1 = \dot{\epsilon}(t) \quad ,$$

the state equations describing the second order phase locked loop become

$$\dot{y}_o = y_1 \quad , \qquad (5\text{-}20)$$

and

$$\dot{y}_1 = -AK\sin(y_1 + ay_o) - K\eta' \quad .$$

The two-dimensional Fokker-Planck equation which describes the joint probability density function of the state variables defined in (5-20) is

$$f_t(y_o,y_1,t) = -y_1 f_{y_o}(y_o,y_1,t) \qquad (5\text{-}21)$$

$$+ AK(\sin(y_1 + y_o)\, f(y_o,y_1,t))_{y_1}$$

$$+ \frac{K^2N_o}{4}\, f_{y_1y_1}(y_o,y_1,t) \quad .$$

This Fokker-Planck equation is put in terms of the phase error of the system, ϕ, by relating the phase error to the state variables. This relationship, given by (5-16) and (5-19), is

$$\phi = y_1 + ay_o \quad . \tag{5-22}$$

Let

$$z = ay_o \quad . \tag{5-23}$$

When these changes of variables are put into (5-21) it becomes

$$\begin{aligned} f_t(\phi,z,t) &= a(z - \phi)(f_\phi(\phi,z,t) + f_z(\phi,z,t)) \\ &+ AK(\sin(\phi)\ f(\phi,z,t))_\phi \\ &+ \frac{K^2N_o}{4}\ f_{\phi\phi}(\phi,z,t) \quad . \end{aligned} \tag{5-24}$$

The magnitude of the integrator gain, "a," must be large in order for it to have a real effect on the system. More precisely, if "a" is small compared to AK, the integrator has little influence on the loop [8]. Under this condition the system behaves like a first order phase-locked loop. Therefore, let

$$a = AK \quad . \tag{5-25}$$

Letting

$$\tau = AKt \quad , \tag{5-26}$$

and using (5-25), (5-24) becomes

$$f_\tau(\phi,z,\tau) = (z - \phi)(f_\phi(\phi,z,\tau) + f_z(\phi,z,\tau)) \quad (5\text{-}27)$$

$$+ (\sin(\phi)\ f(\phi,z,\tau))_\phi$$

$$+ \frac{KN_o}{4A}\ f_{\phi\phi}(\phi,z,\tau) \quad .$$

The signal to noise ratio in the second order loop is defined as [8]

$$\alpha = \frac{A^2}{N_o B_1} = \frac{A^2}{N_o \frac{(AK+a)}{4}} = \frac{2A}{N_o K} \quad , \quad (5\text{-}28)$$

where B_1 is the loop noise bandwidth for the linearized system. Therefore, the Fokker-Planck equation for the second order phase-locked loop becomes

$$L(\phi,z,\tau) = - f_\tau(\phi,z,\tau) + (z - \phi)(f_\phi(\phi,z,\tau) + f_z(\phi,z,\tau)) \quad (5\text{-}29)$$

$$+ (\sin(\phi)\ f(\phi,z,\tau))_\phi + \frac{1}{2\alpha} f_{\phi\phi}(\phi,z,\tau) = 0 \quad .$$

This equation is the desired final form of the Fokker-Planck equation in terms of the joint probability density function.

At this point, a few remarks should be made about the system being considered. Some of the physical characteristics of phase-locked loops are defined in terms of the linearized form of the system (the loop becomes a linear system in the region where $\sin(\phi) \simeq \phi$). The differential equation describing the operation of (5-15) in the linear region when there is no noise input is

$$\ddot{\phi} + AK\dot{\phi} + aAK\phi = 0 \tag{5-30}$$

Using the changes of variables defined by (5-25) and (5-26), (5-30) becomes

$$\ddot{\phi} + \dot{\phi} + \phi = 0 \quad , \tag{5-31}$$

where the derivatives are now with respect to the normalized time variable τ. One important parameter to consider in the operation of a second order phase-locked loop is the damping coefficient of the system. The damping coefficient of the nonlinear system is interpreted as that of the linear model. Therefore, for the system being considered, the damping coefficient is

$$\delta = 0.5 \quad . \tag{5-32}$$

This is a very reasonable value for the second order system. Therefore, the selection of a = AK was a good choice. Different damping coefficients can be obtained by selecting different values for the integrator gain. For example, if $a = \frac{AK}{4}$ the damping coefficient becomes one.

It is desired to obtain the solution on modulo 2π. Since the phase error, ϕ, is always interpreted as being between $+\pi$ and $-\pi$, it is clear from (5-16) and (5-23) that the other independent space variable, z, is also always between $+\pi$ and $-\pi$.

Again, before any solutions can be sought, it is necessary to establish adequate initial and boundary conditions. No apriori knowledge is assumed about the phase error. Therefore, the initial condition is a uniform density function. The boundary conditions involve the values of the probability current densities on the boundaries.

Due to the symmetry of this problem it is again clear that the probability current on the boundary is zero (the probability current across a boundary is the integral of the probability current density on the boundary). The initial conditions which are used are that the probability current densities on the boundaries are zero. That is

$$J_\phi(-\pi,z,\tau) = J_\phi(\pi,z,\tau) = 0 \quad , \tag{5-33}$$

$$J_z(\phi,-\pi,\tau) = J_z(\phi,\pi,\tau) = 0 \quad ,$$

where

$$J_\phi(\phi,z,\tau) = (z - \phi + \sin(\phi))\; f(\phi,z,\tau) + \frac{1}{2\alpha}\, f_\phi(\phi,z,\tau), \tag{5-34}$$

$$J_z(\phi,z,\tau) = (z - \phi)\; f(\phi,z,\tau) \quad .$$

The quantities $J_\phi(\phi,z,\tau)$ and $J_z(\phi,z,\tau)$ are the probability current densities in the ϕ and z directions respectively. Therefore, the problem to be solved in terms of the probability density function is

$$f_\tau(\phi,z,\tau) = (z - \phi)\;(f_\phi(\phi,z,\tau) + f_z(\phi,z,\tau)) \tag{5-35}$$

$$+ (\sin(\phi)\; f(\phi,z,\tau))_\phi + \frac{1}{2\alpha}\, f_{\phi\phi}(\phi,z,\tau) \quad ,$$

$$f(\phi,z,0) = 1/4\pi^2 , \quad -\pi \le \phi \le \pi \quad \text{and} \quad -\pi \le z \le \pi$$

$$J_\phi(-\pi,z,\tau) = 0 ,$$

$$J_\phi(\pi,z,\tau) = 0 ,$$

$$J_z(\phi,-\pi,\tau) = 0 ,$$

and

$$J_z(\phi,\pi,\tau) = 0 .$$

Although it is clear that the probability currents at the boundaries are zero this does not necessarily mean that the probability current densities on the boundaries are also zero. Therefore, a few comments are in order about the possible physical interpretations of these boundary conditions. There seem to be two possible physical interpretations. The first is that this representation is the modulo 2π problem. However, Lindsey suggests that the solution to (5-35) is the density function of the collection of trajectors which were initially uniformly distributed within the prime interval and which have remained strictly within the interval. That is, starting with an ensemble uniformly distributed on $-\pi \le \phi \le \pi$ and $-\pi \le z \le \pi$, (5-35) describes the density function of the sample paths which have remained strictly within the boundaries. According to this interpretation any trajectories which reach the boundary are removed from the ensemble (Figure 24). There are physical arguments supporting either interpretation. These two interpretations appear to be very similar and the results of each should be very close. Therefore, whichever is the true interpretation, the results are a good indication of the phase error of the second order loop.

In order to obtain numerical results it is desirable to transform (5-35) into an equivalent equation in terms of the probability distribution function. In order to obtain this formulation (5-35) is integrated with respect to its two independent space variables. That is, referring to (5-29) ,

$$\int_{\pi}^{z} \int_{-\pi}^{\phi} L(\epsilon,\xi,\tau)d\epsilon d\xi = 0 \quad . \tag{5-36}$$

The resulting equation is

$$\begin{aligned} F_\tau(\phi,z,\tau) &= (z - \phi + \sin(\phi)) \quad F_\phi(\phi,z,\tau) \\ &+ (z - \phi) \; F_z(\phi,z,\tau) + \frac{1}{2\alpha} F_{\phi\phi}(\phi,z,\tau) \\ &+ \int_{-\pi}^{\phi} F_z(\epsilon,z,\tau) \; d\epsilon - \int_{-\pi}^{z} F_\phi(\phi,\epsilon,\tau) \; d\epsilon \\ &- \int_{-\pi}^{\phi} J_z(\epsilon,-\pi,\tau) \; d\epsilon - \int_{-\pi}^{z} J_\phi(-\pi,\epsilon,\tau) \; d\epsilon \quad . \end{aligned} \tag{5-37}$$

Using (5-33), (5-37) becomes

$$\begin{aligned} F_\tau(\phi,z,\tau) &= (z - \phi + \sin(\phi)) \; F_\phi(\phi,z,\tau) \\ &+ (z - \phi) \; F_z(\phi,z,\tau) + \frac{1}{2\alpha} \; F_{\phi\phi}(\phi,z,\tau) \quad \text{(continued)} \end{aligned} \tag{5-38}$$

$$+ \int_{-\pi}^{\phi} F_z(\epsilon,z,\tau)\, d\epsilon - \int_{-\pi}^{z} F_\phi(\phi,\epsilon,\tau)\, d\epsilon \quad .$$

The initial condition and half of the boundary conditions clearly become

$$F(\phi,z,0) = \frac{(\phi+\pi)(z+\pi)}{4\pi^2} , \quad -\pi \le \phi \le \pi \ \text{ and } -\pi \le z \le \pi \tag{5-39}$$

$$F(-\pi,z,\tau) = 0 \quad ,$$

and

$$F(\phi,-\pi,\tau) = 0 \quad .$$

The boundary conditions at $\phi = \pi$ and $z = \pi$ have to be obtained from (5-33). In particular, since $J_\phi(\pi,z,\tau) = J_z(\phi,\pi,\tau) = 0$

$$\int_{-\pi}^{z} J_\phi(\pi,\epsilon,\tau)\, d\epsilon = \Big[(z - \phi + \sin\phi)\, F_\phi(\phi,z,\tau) \tag{5-40}$$

$$+ \frac{1}{2\alpha}\ F_{\phi\phi}(\phi,z,\tau) - \int_{-\pi}^{\phi} F_\phi(\phi,\epsilon,\tau)\, d\epsilon\Big]_{\phi=\pi} = 0 \ ,$$

$$\int_{-\pi}^{\phi} J_z(\epsilon,\pi,\tau)\, d\epsilon = \Big[(z-\phi)\, F_z(\phi,z,\tau) + \int_{-\pi}^{\phi} F_z(\epsilon,z,\tau)\, d\epsilon\Big]_{z=\pi} = 0 \ .$$

In summary, the equation to be simulated is

$$F_\tau = (z - \phi + \sin\phi)\, F_\phi + (z - \phi)\, F_z \tag{5-41}$$

$$+ \frac{1}{2\alpha} F_{\phi\phi} + \int_{-\pi}^{\phi} F_z d\phi - \int_{-\pi}^{z} F_\phi dz \quad ,$$

$$F(\phi,z,0) = \frac{(\phi+\pi)(z+\pi)}{4\pi^2} \quad , \quad -\pi \le \phi \le \pi \ \text{ and } \ -\pi \le z \le \pi$$

(continued)

$$F(-\pi, z, \tau) = 0 \quad ,$$

$$F(\phi, -\pi, \tau) = 0 \quad ,$$

$$\left[(z - \phi + \sin \phi)\, F_{\phi} + \frac{1}{2\alpha}\, F_{\phi\phi} - \int_{-\pi}^{z} F_{\phi} dz \right]_{\phi = \pi} = 0 \quad ,$$

and

$$\left[(z - \phi)\, F_{z} + \int_{-\pi}^{\phi} F_{z} d\phi \right]_{z = \pi} = 0 \quad .$$

As was the case for the Fokker-Planck equation for the second order linear system the step sizes used in the numerical solutions of (5-41) can be larger than those suggested by the maximum principle. For the most part the step sizes used were $\Delta\phi = \Delta z = \pi/50$ and $\Delta\tau = 0.001$ (for complete details see Appendix B).

The modified algorithm was used to solve (5-41) for signal to noise ratios of 0.41 (-3.87 db), 1.1 (0.41 db), 1.382 (1.41 db), and 2.76 (4.41 db). The results which were obtained are more striking when presented in terms of the probability density function. Therefore, the solutions are differentiated in order that they can be plotted both in terms of density and distribution functions. These results are shown in Figures 31 through 38 (the details of these simulations are tabulated in Appendix B).

These results are somewhat surprising in that the density functions display a multimodel structure. Lindsey and Charles [26] obtained some experimental steady state density and distribution functions for the phase error of a second order phase-locked loop. Their results display this same multimodel structure for low signal to noise ratios. They also

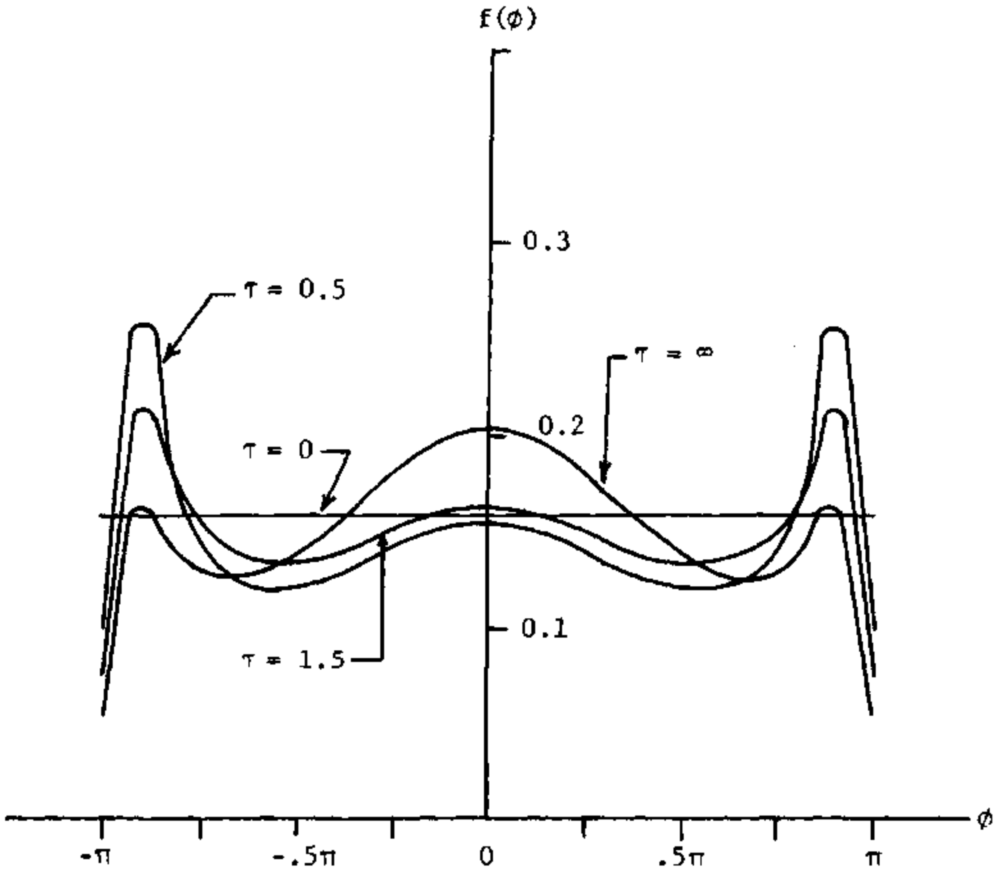


Figure 31. Density Function of the Phase Error for the Second Order Phase-Locked Loop-- $\alpha = 0.41$

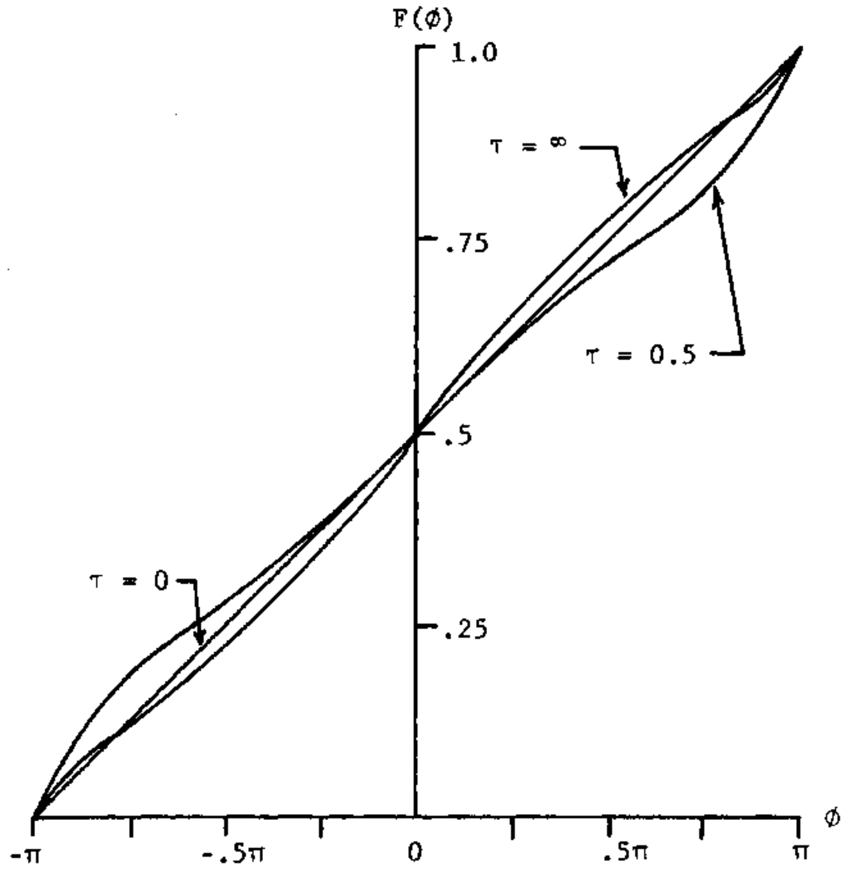


Figure 32. Distribution Function of the Phase Error for the Second Order Phase-Locked Loop-- $\sigma = 0.41$

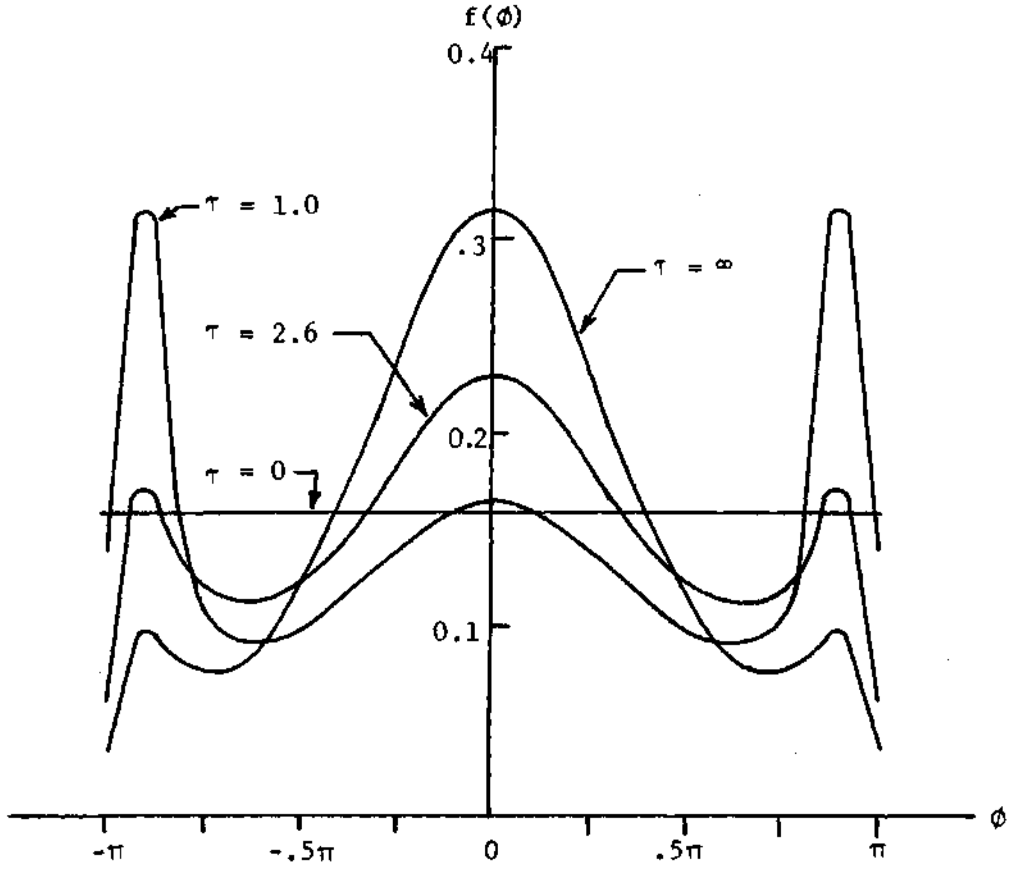


Figure 33. Density Function of the Phase Error for the Second Order Phase-Locked Loop--α = 1.1

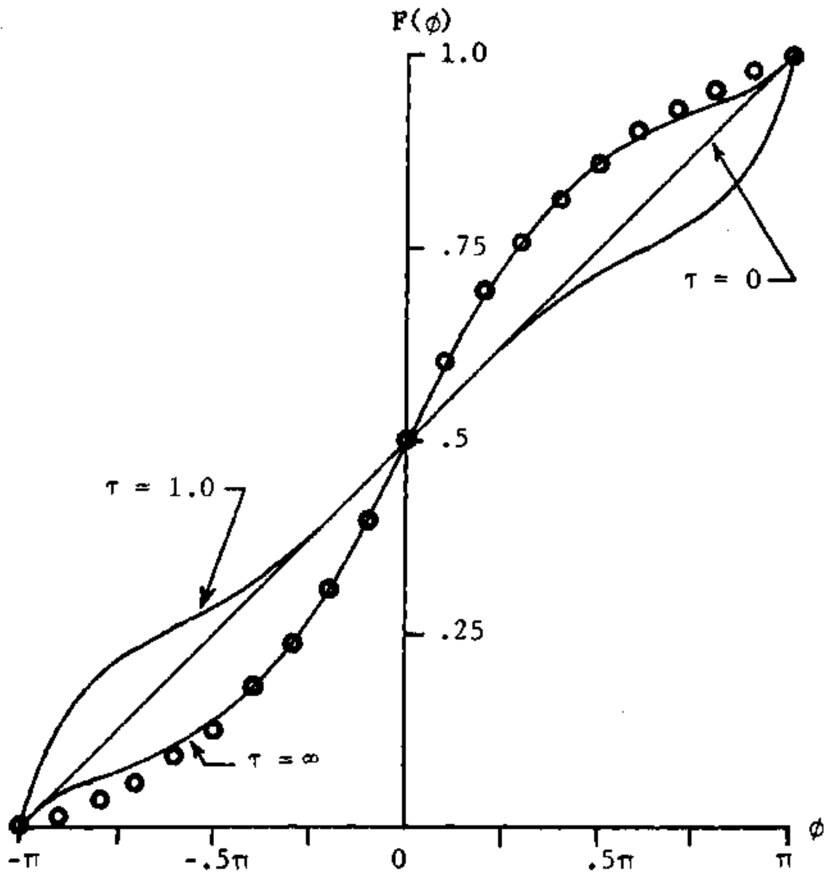


Figure 34. Distribution Function of the Phase Error for the Second Order Phase-Locked Loop-- $\sigma = 1.1$. (The circles are Lindsey and Charles' experimental steady state results for a second order system with a damping ratio that is slightly different than that which was used for the numerical calculations. See the text.)

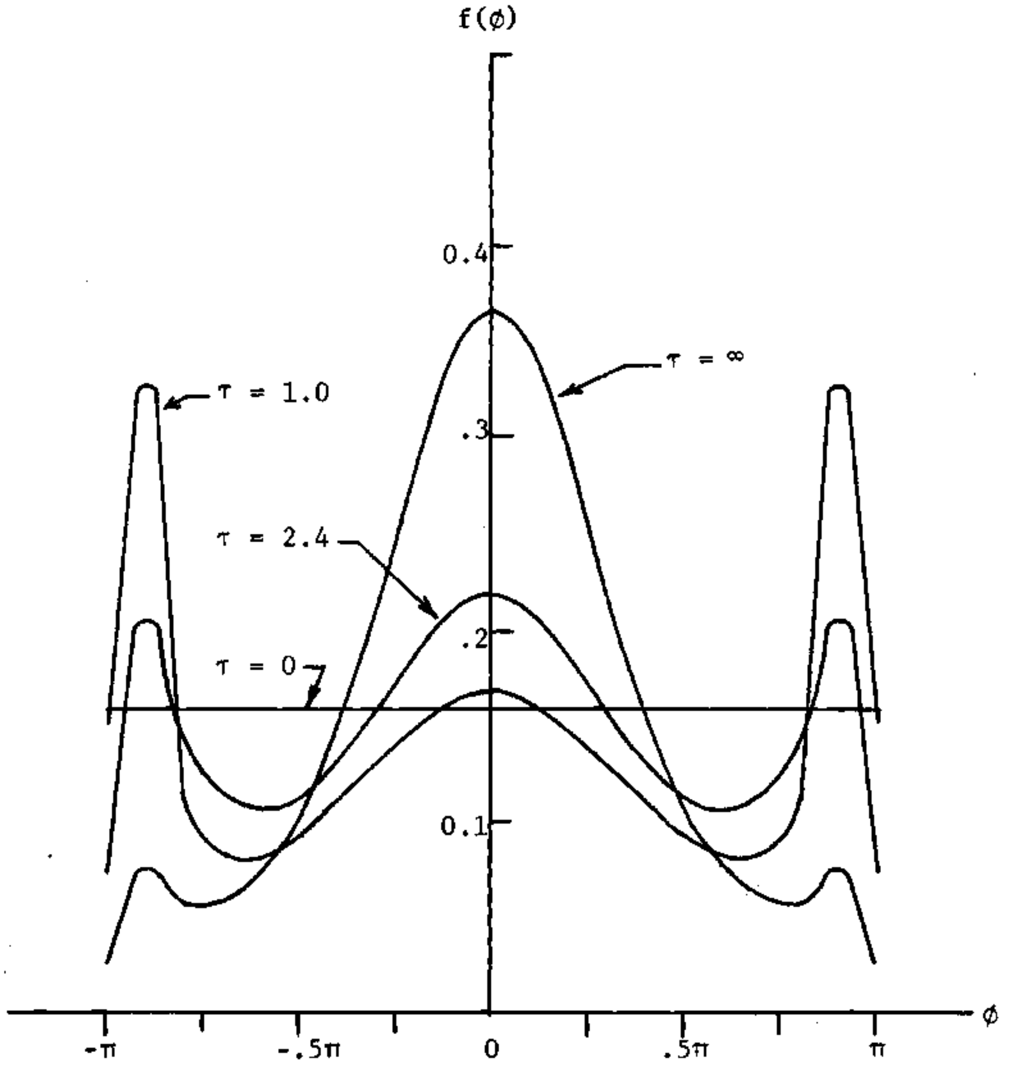


Figure 35. Density Function of the Phase Error for the Second Order Phase-Locked Loop--α = 1.382

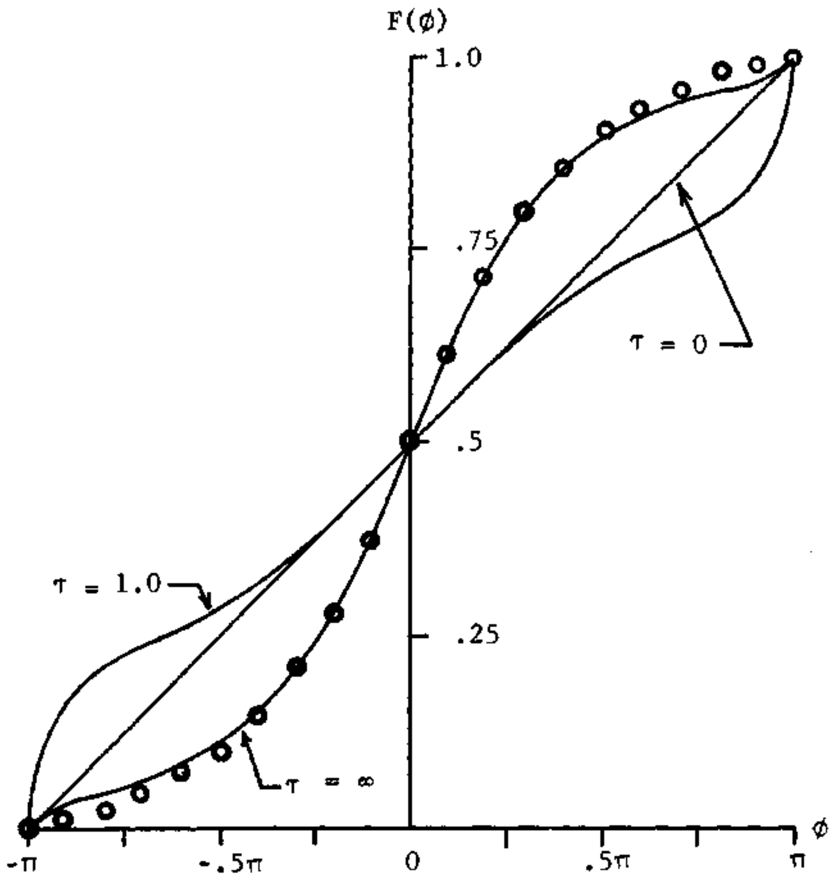


Figure 36. Distributed Function of the Phase Error for the Second Order Phase-Locked Loop-- $\alpha = 1.382$. (The circles are Lindsey and Charles' experimental steady state results for a second order system with a damping ratio that is slightly different than that which was used for the numerical calculations. See the text.)

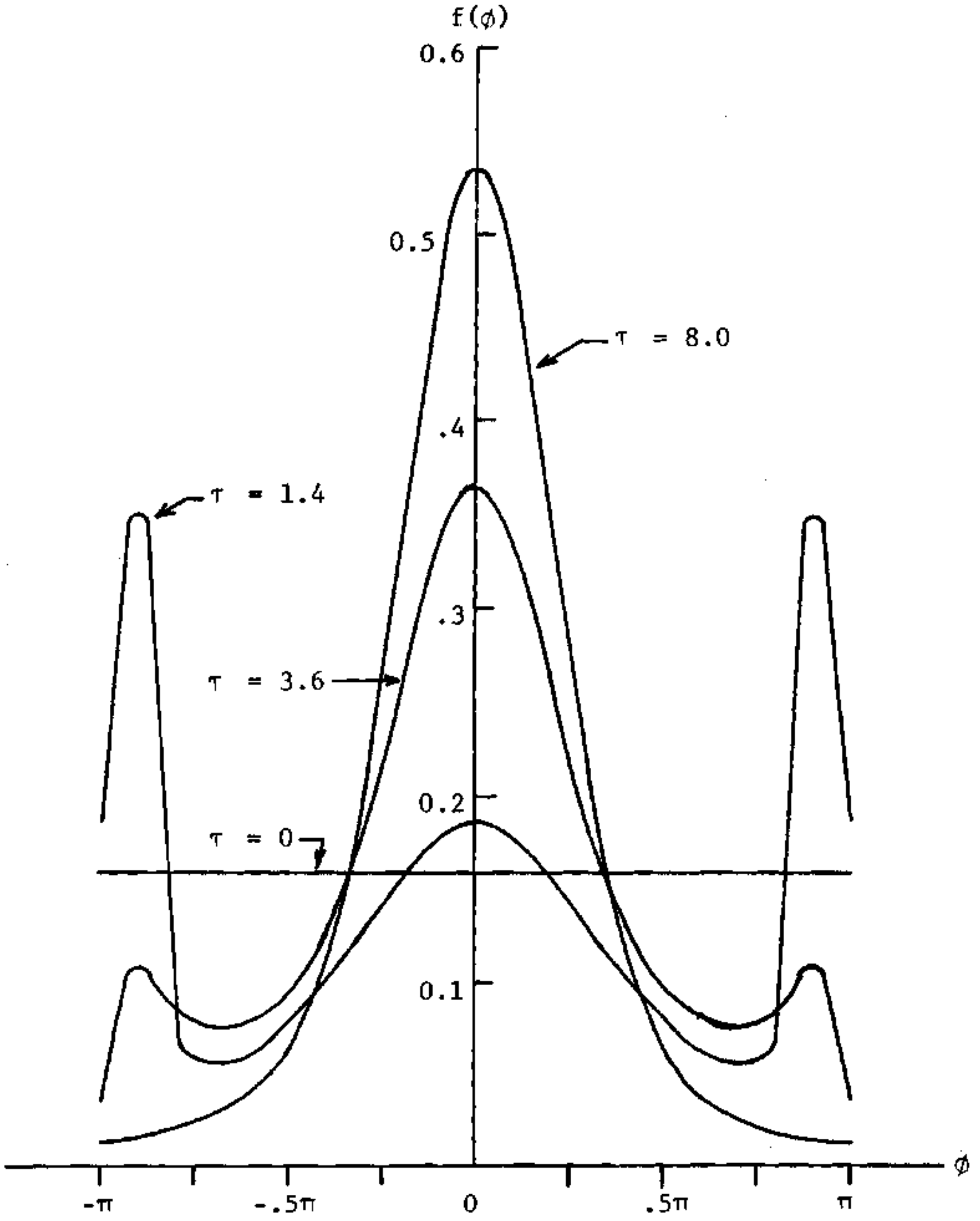


Figure 37. Density Function of the Phase Error for the Second Order Phase-Locked Loop-- $\alpha = 2.76$

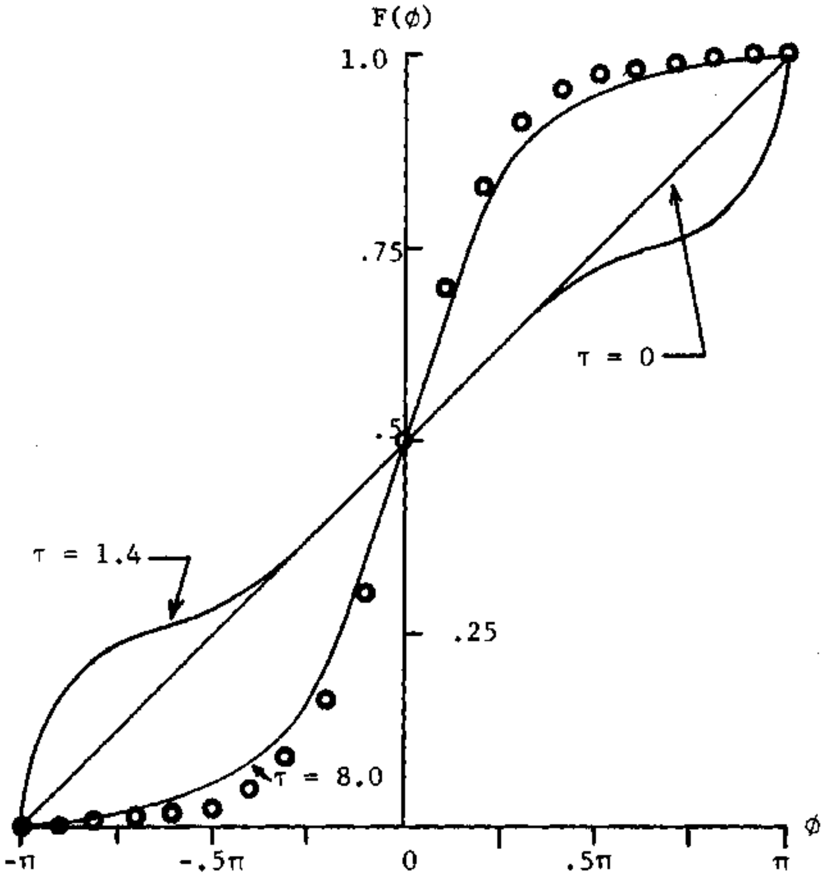


Figure 38. Distribution Function of the Phase Error for the Second Order Phase-Locked Loop-- $\alpha = 2.76$. (The circles are Lindsey and Charles' experimental steady state results for a second order system with a damping ratio that is slightly different than that which was used for the numerical calculations. See the text.)

state that the mathematical analysis of the problem gives an indication of such a structure.

Although several of the solutions obtained with the modified algorithm were for systems with the same signal to noise ratios as used by Lindsey and Charles [26] to obtain their experimental solutions, the steady state results of the two methods are not directly comparable. The damping ratios of the systems for which the experimental and numerical results were obtained are 0.707 and 0.5 respectively. However, the two solutions should be, and are, close. The steady state distribution functions for the two systems with the same signal to noise ratios are very close, differing only slightly in the regions of low probability.

The variances as a function of time for two of the previous runs are plotted in Figure 39. These graphs are for signal to noise ratios of 1.1 (0.41 db) and 1.382 (1.41 db). The reason that only two of the variances are plotted is that initially the variances were not obtained, and it was necessary to completely rerun the problems in order to get them. This involved so much computer time that only the variances for the two middle signal to noise ratios were obtained.

Lindsey [41] presents a graph of approximate theoretical steady state variances as functions of signal to noise ratios and system damping ratios for a second order loop. The steady state values for the two variances shown in Figure 39 are consistent with the curves and data Lindsey shows.

Gated Phase-Locked Loop

In this section the modified algorithm is used to solve the

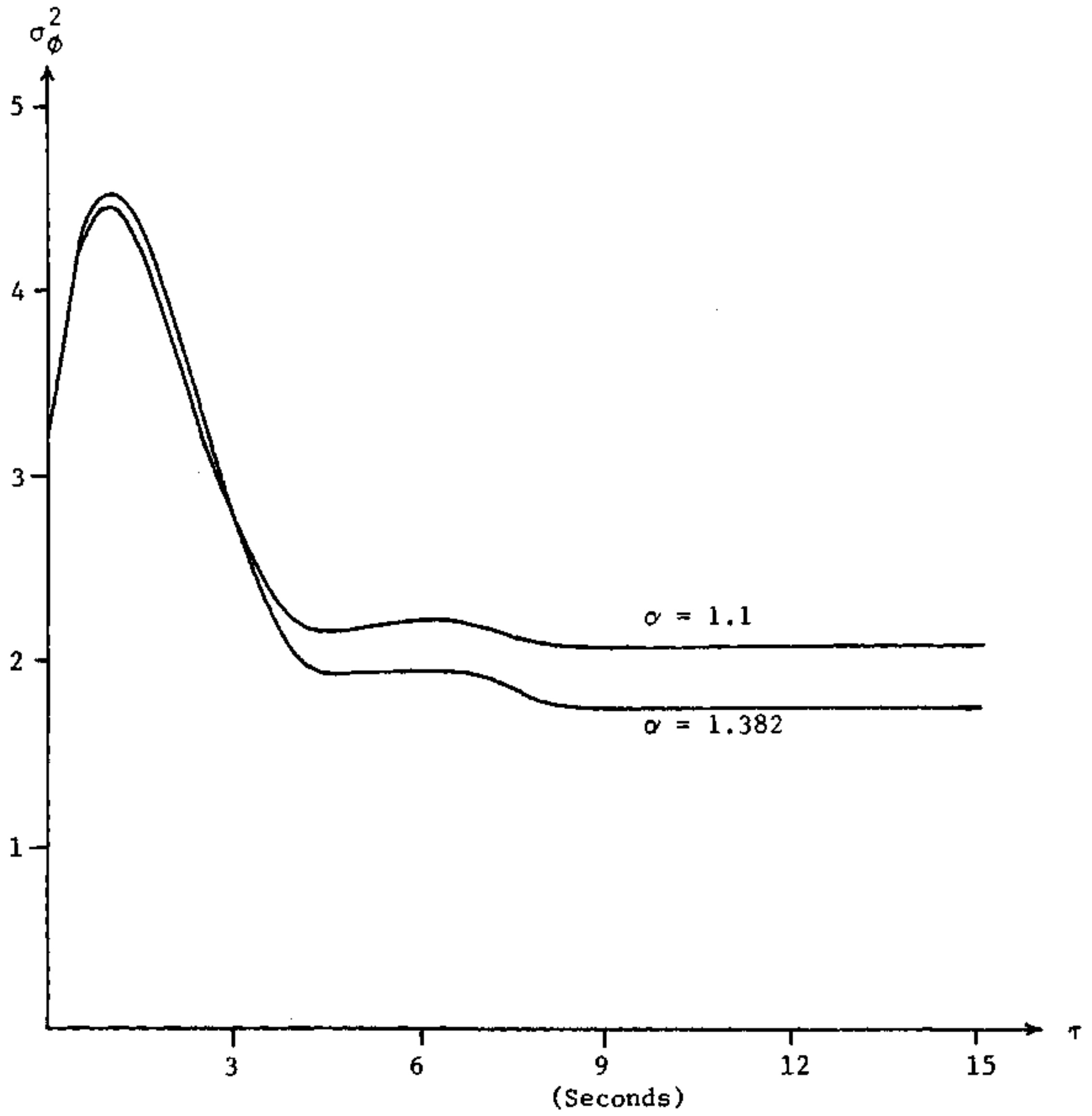


Figure 39. Variances of the Phase-Errors for the Second Order Phase-Locked Loop--$\alpha = 1.1$, $\alpha = 1.382$.

Fokker-Planck equations for a first and second order grated phase-locked loop [46] [47]. This problem arises in a Time Division Multiple Access (TDMA) system which uses Phase Shift Keyed (PSK) modulation and which requires that phase coherence be maintained from burst to burst. The data modulation on the carrier is assumed to be removed by the demodulator and only the problem of carrier tracking is considered. The variance of the phase error of the carrier is obtained and plotted for several different signal to noise ratios

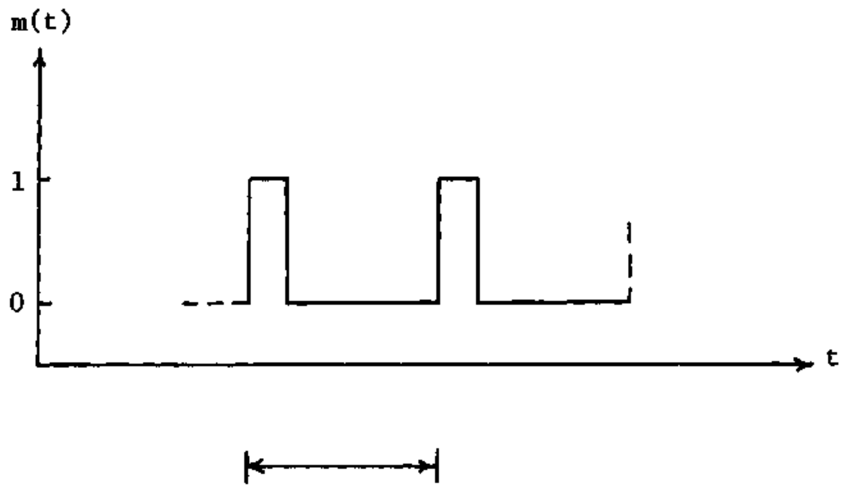
The problem is setup the same as in the previous sections except that now the input is

$$\sqrt{2}\, A\, m(t)\, \cos(\omega t) + \eta(t) \quad , \tag{5-42}$$

where m(t) is a periodic gate function (Figure 40). The Fokker-Planck equation, in terms of the distribution function, for the first order phase-locked loop when the gate is on (m(t) = 1) is given by (5-9). During that portion of the time frame when the gate is off (m(t) = 0) the Fokker-Planck equation becomes

$$F_{\tau}(\phi,\tau) = \frac{1}{\alpha} F_{\phi\phi}(\phi,\tau) \quad . \tag{5-43}$$

Paul and Larimore [47] state that the assumption of burst to burst coherence means that the time constant of the phase-locked loop is larger than the time frame of the gated system (period of the gate function). The time constant of the loop is interpreted as that of the linearized system. In terms of the normalized time variable, τ, the time constant of the first order loop is unity. The time frame selected



TDMA Time Frame

Figure 40. Gate Function for a TDMA System

for this problem (in terms of the normalized time) is 0.5.

The problem was initially solved (see Appendix B for details) with the time frame having a duty factor of 0.1 (m(t) = 1 one-tenth of the time). It was observed that the steady state variances of the phase errors of the gated loop were the same as those for the continuous loop with signal to noise ratios of ten times those of the gated loop. In other words, it was observed that steady state variances of the phase errors for the two loops were the same if the systems received the same amount of input signal energy per time frame. Figure 41 shows the computed steady state variances for the gated system with a duty factor of 0.1 and for signal to noise ratios of 2, 5, 10, 20, 30, and 40. The computed steady state variances for the continuous system with signal to noise ratios of 0.1, 1, 2, 3, and 4 are also shown.

These results indicate that the solution for the continuous system can be used to obtain the steady state variance for a gated first order phase-locked loop. This is an interesting and helpful result since the solution to the continuous system requires far less computer time than the solution for the gated loop.

One further run was made to check this apparent relationship between the steady state variances of the phase errors of the two systems. Figure 42 shows the steady state variances computed for the phase errors of the first order gated phase-locked loop with a duty factor of 0.05 and for signal to noise ratios of 10, 20, and 40. The steady state variances for the continuous loop are also plotted for signal to noise ratios of 0.5, 1, and 2. Again the steady state variances for the continuous and gated cases are the same if the input signal energy per

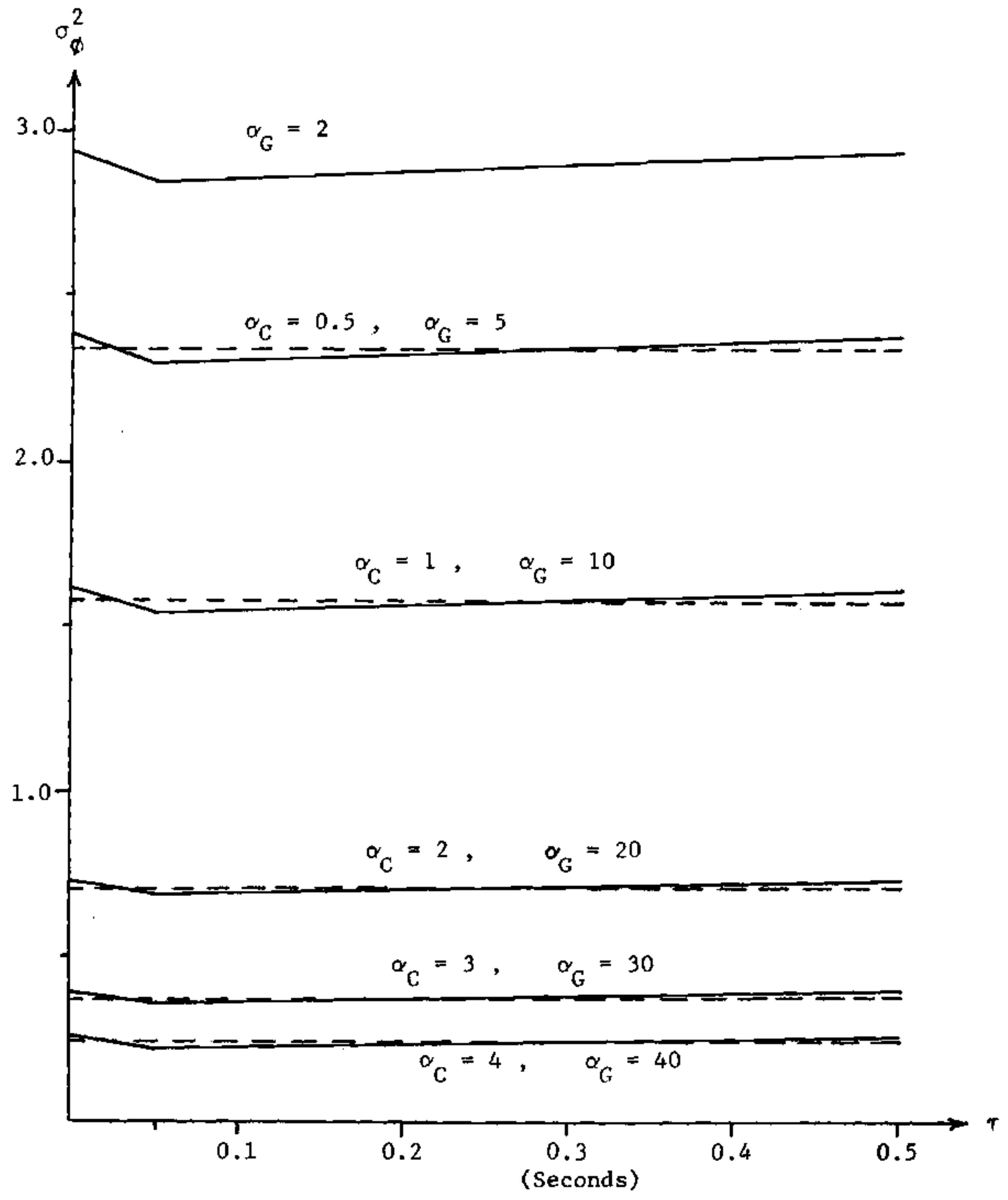


Figure 41. Comparison of the Steady State Variances of the Phase Errors for the First Order Gated and Continuous Phase-Locked Loops--Duty Factor = 0.1 (The solid curves are for the gated loop and the dashed curves are for the continuous loop.)

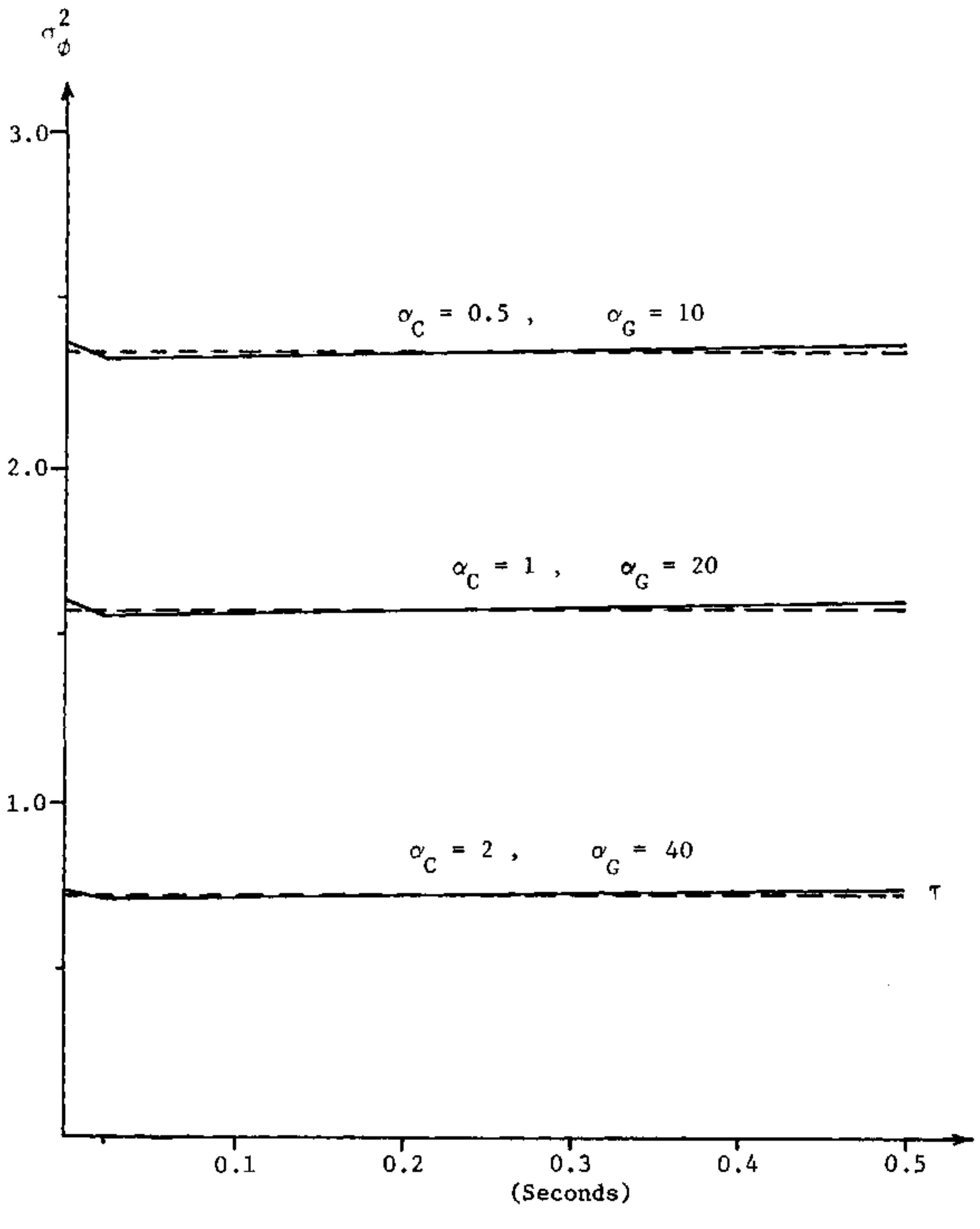


Figure 42. Comparison of the Steady State Variances of the Phase Errors for the First Order Gated and Continuous Phase-Locked Loops--Duty Factor = 0.05 (The solid curves are for the gated loop and the dashed curves are for the continuous loop.)

time frame are the same for the two systems.

The computer times required for the solutions of the Fokker-Planck equations for the gated and continuous loops to reach steady state were examined to see if there was a direct relationship between them. Although it did take much more time for the gated loop to reach steady state, the two times do not appear to adhere to a formula. In general, for a duty factor of $\frac{1}{N}$, the time required for the gated loop (with signal to noise ratio $N\alpha$) to reach steady state was somewhat less than N times the time required for the continuous system(with signal to noise ratio α) to reach steady state.

The second order gated phase-locked loop is setup as in the previous section except that the input to the system is given by (5-42). The Fokker-Planck equation, in terms of the distribution function, for the system while the gate function is on ($m(t) = 1$) is given by (5-41). During that portion of the time frame when the gate function is off ($m(t) = 0$) the Fokker-Planck equation becomes

$$\begin{aligned} F_\tau(\phi,z,\tau) &= (z - \phi)\, F_\phi(\phi,z,\tau) \qquad (5\text{-}44) \\ &+ (z - \phi)\, F_z(\phi,z,\tau) + \frac{1}{2\alpha} F_{\phi\phi}(\phi,z,\tau) \\ &+ \int_{-\pi}^{\phi} F_z(\epsilon,z,\tau)\, d\epsilon - \int_{-\pi}^{z} F_\phi(\phi,\epsilon,\tau)\, d\epsilon \quad . \end{aligned}$$

Unless initial conditions can be obtained which are close to the steady state solutions it is anticipated that the solutions to the Fokker-Planck equation for the second order gated phase-locked loop will require very large amounts of computer time. Therefore, the second order gated and continuous loops are investigated to see if a relationship similar to the one observed for the first order loops exists. However, it is found (from several simulations) that the desired relationship does not hold for the second order systems.

The problem is solved starting from an initial distribution which is uniformly distributed. The time frame is again 0.5. For these simulations the variances of the phase errors at the beginning of each time frame vs. the number of time frames is plotted. Figure 43 shows the results obtained for the second order gated loop with a duty factor of 0.5 and signal to noise ratios of 2.2 and 4. Figure 44 shows the results obtained for a duty factor of 0.25 and for signal to noise ratios of 6 and 9.

These solution, which use large duty factors and fairly small signal to noise ratios, require large amounts of computer time (the longest run took about 37 minutes of computer time). Even with the large savings in computer time realized with the modified algorithm it is clear that the computer time required to obtain a complete solution to the Fokker-Planck equation for a second order gated loop with a practical duty factor is completely prohibitive.

However, the variance of the phase error for the second order gated system demonstrates a pattern which can be used to get a good estimate of the steady state value without obtaining the complete solution to the problem. The plots of the variances for both the continuous and gated second order loops are very similar in shape. The variances initially start at $\pi^2/3$, rise to a peak value, decrease, and then

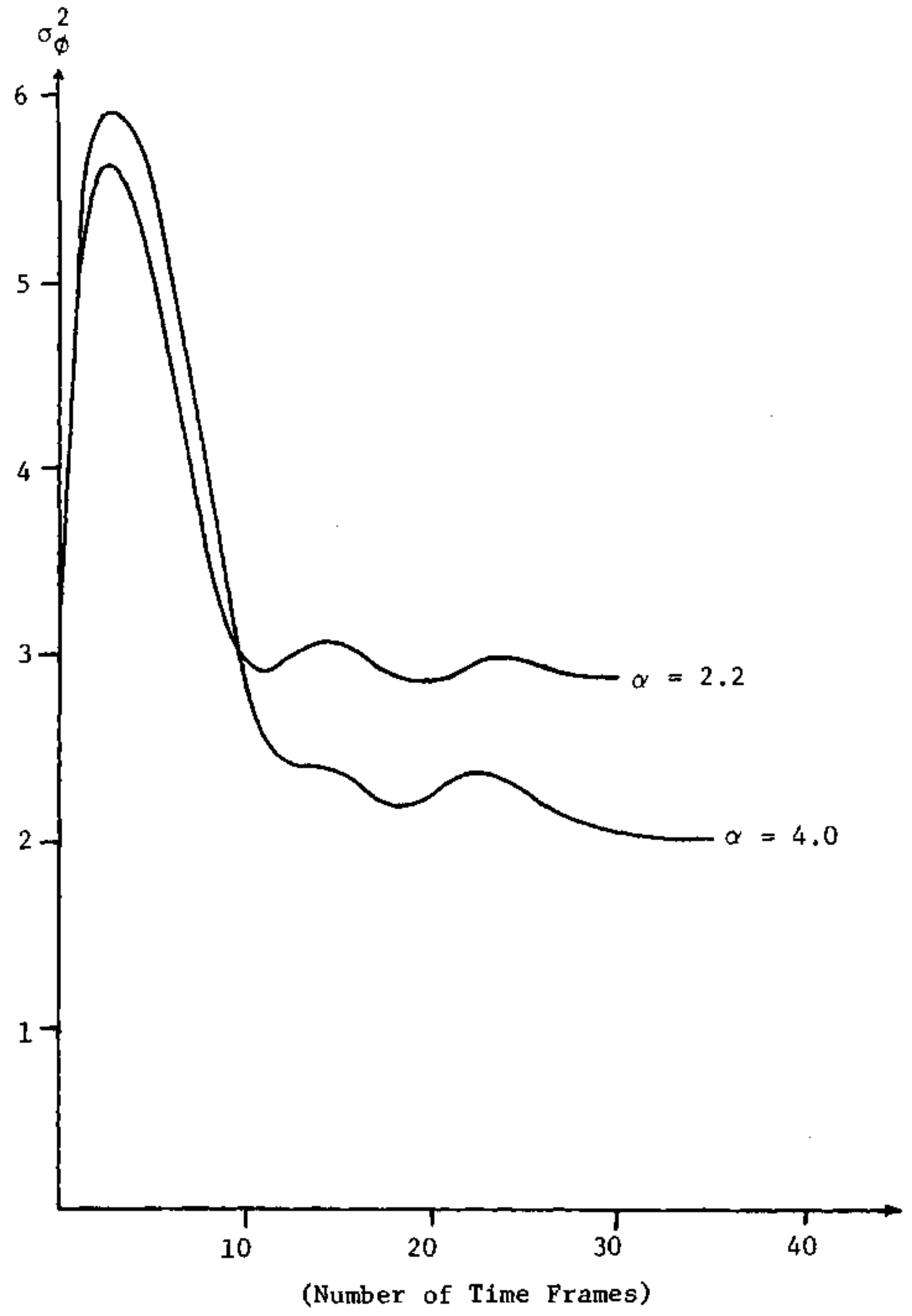


Figure 43. Variances of the Phase Errors for the Second Order Gated Phase-Locked Loop--Duty Factor = 0.5

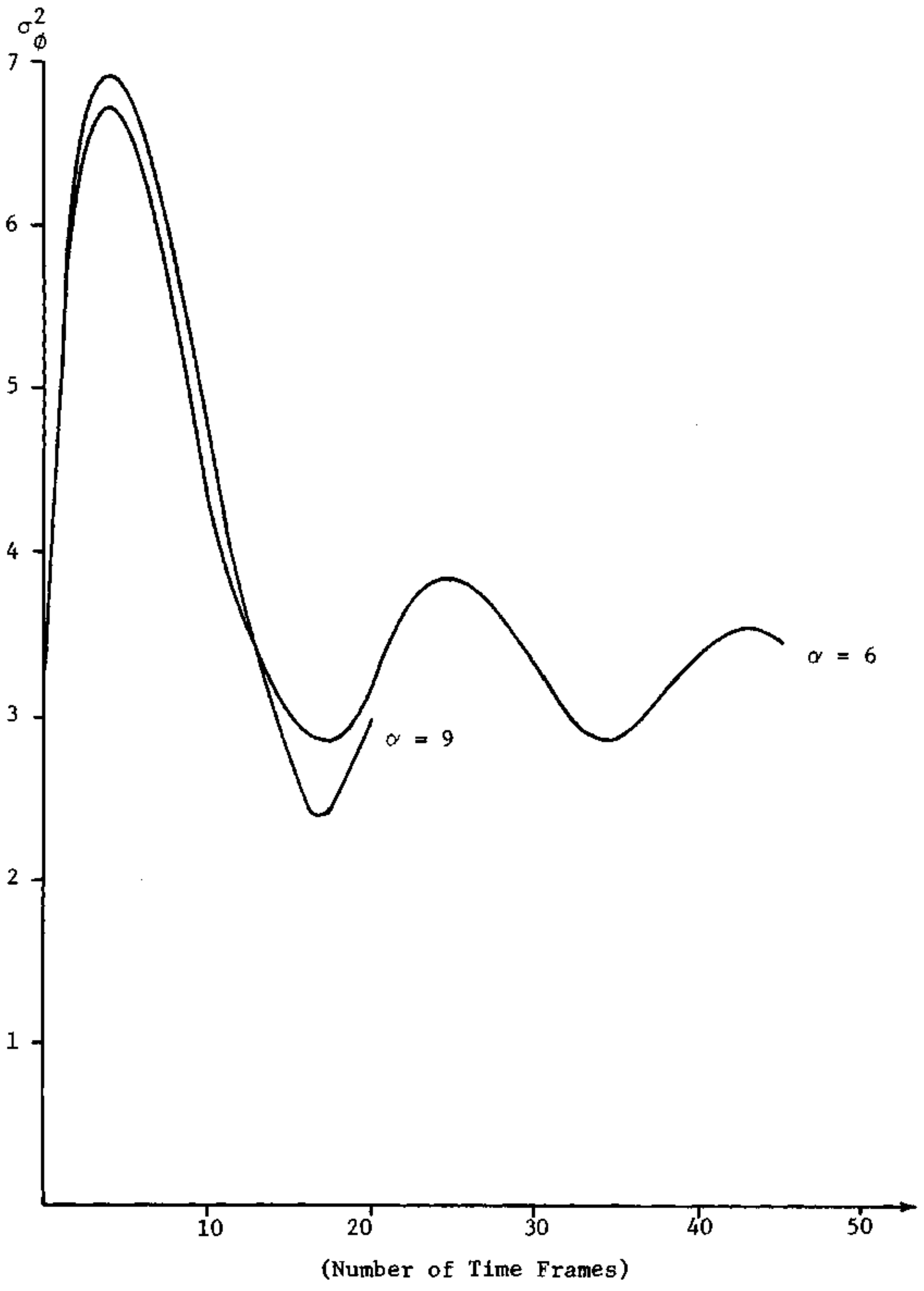


Figure 44. Variances of the Phase Errors for the Second Order Gated Phase-Locked Loop--Duty Factor = 0.25

begin a decaying oscillation to steady state. Notice that the steady state value for the variance is very close to the value of the first distinct minimum in the plot.

Recognizing this pattern it is possible to estimate the steady state variance of the phase error without using tremendous amounts of computer time. For example, from Figure 44, the steady state variance for the second order gated loop with a duty factor of 0.25 and a signal to noise ratio of 9 can be estimated to be about 2.4.

CHAPTER VI

THREE-DIMENSIONAL FOKKER-PLANCK EQUATION

The amounts of computer time and storage required for standard numerical solutions of higher-dimensional parabolic partial differential equations are so enormous that such solutions have not been feasible. The modified algorithm can be used to reduce the amounts of computer time and storage required for such problems to the point where some higher-dimensional equations can be solved quite easily. In this chapter the modified algorithm is used to solve a three-dimensional Fokker-Planck equation.

The partial differential equation considered is the Fokker-Planck equation for the third order system illustrated in Figure 45. The three inputs to the system are identical, but independent, Gaussian white noises with power spectral heights of four. The state equations for the system are

$$\dot{x} = - x + \eta_1(t) \ , \tag{6-1}$$

$$\dot{y} = - y + \eta_2(t) \ ,$$

and

$$\dot{z} = - z + \eta_3(t) \ .$$

The Fokker-Planck equation which describes the joint probability density function of the states of the system is

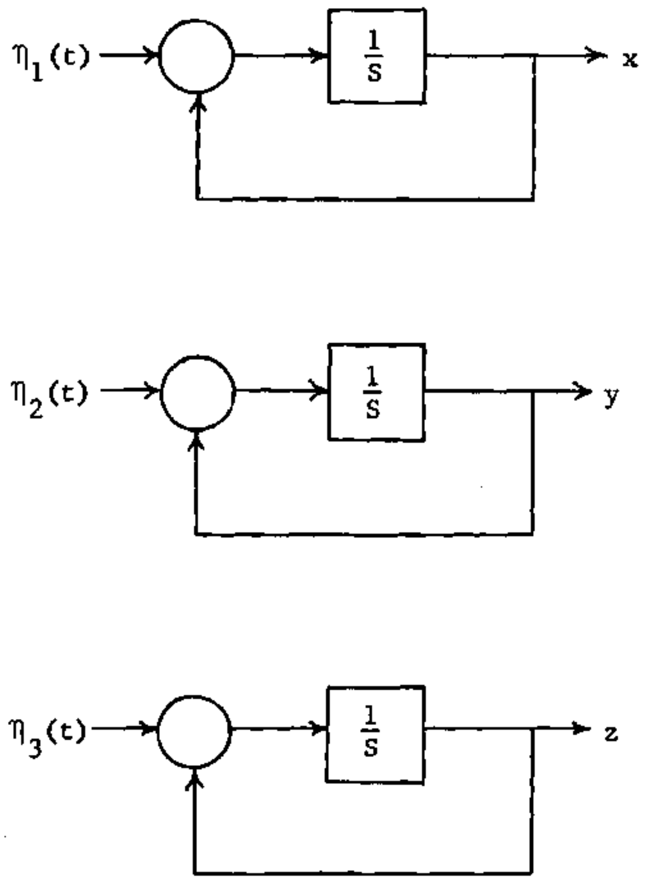


Figure 45. Third Order Linear System

$$L(x,y,z,t) = - f_t(x,y,z,t) + (xf(x,y,z,t))_x + f_{xx}(x,y,z,t) \qquad (6\text{-}2)$$
$$+ (yf(x,y,z,t))_y + f_{yy}(x,y,z,t) + (zf(x,y,z,t))_z + f_{zz}(x,y,z,t) = 0 \ .$$

Again it is desirable to change (6-2) to an equivalent equation in terms of the probability distribution function. This is accomplished by integrating (6-2) with respect to its three independent space variables. That is

$$\int_{-\infty}^{z}\int_{-\infty}^{y}\int_{-\infty}^{x} L(\delta,\epsilon,\xi,t)d\delta d\epsilon d\xi \ . \qquad (6\text{-}3)$$

The resulting equation in terms of the joint probability distribution function of the states of the system is

$$F_t(x,y,z,t) = xF_x(x,y,z,t) + F_{xx}(x,y,z,t) + yF_y(x,y,z,t) \qquad (6\text{-}4)$$
$$+ F_{yy}(x,y,z,t) + zF_z(x,y,z,t) + F_{zz}(x,y,z,t) \ .$$

Defining notation similar to that used previously, let

$$F(x_i,y_j,z_K,t_n) = F(x_0+i\Delta x,y_0+j\Delta y,z_0+K\Delta z,n\Delta t) = F^n_{ijK} \ . \qquad (6\text{-}5)$$

Using (6-5), the explicit finite difference equation which approximates (6-4) is

$$F^{n+1}_{ijK} = F^n_{i+1,jK}\left(x_i \frac{\Delta t}{2\Delta x} + \frac{\Delta t}{\Delta x^2}\right) + F^n_{i-1,jK}\left(- x_i \frac{\Delta t}{2\Delta x} + \frac{\Delta t}{\Delta x^2}\right) \qquad (6\text{-}6)$$

(continued)

$$+ F^n_{i,j+1,K}\left(y_j \frac{\Delta t}{2\Delta y} + \frac{\Delta t}{\Delta y^2}\right) + F^n_{i,j-1,K}\left(- y_j \frac{\Delta t}{2\Delta y} + \frac{\Delta t}{\Delta y^2}\right)$$

$$+ F^n_{ij,K+1}\left(z_K \frac{\Delta t}{2\Delta z} + \frac{\Delta t}{2\Delta z^2}\right) + F^n_{ij,K-1}\left(- z_K \frac{\Delta t}{2\Delta z} + \frac{\Delta t}{\Delta z^2}\right)$$

$$+ F^n_{ijK}\left(1 - 2\frac{\Delta t}{\Delta x^2} - 2\frac{\Delta t}{\Delta y^2} - 2\frac{\Delta t}{\Delta z^2}\right) .$$

An easy extension of the maximum principle to three dimensions yields an adequate criterion for the selection of step sizes in order to solve (6-6). For this particular problem the maximum principle becomes

$$\Delta x \le \frac{|x|_{max}}{2} , \qquad (6\text{-}7)$$

$$\Delta y \le \frac{|y|_{max}}{2} ,$$

$$\Delta z \le \frac{|z|_{max}}{2} ,$$

and

$$\Delta t \le \frac{1}{\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2} + \frac{2}{\Delta z^2}} .$$

All of the spatial step sizes will be chosen to be the same value. Therefore, the last equation of (6-7) becomes

$$\Delta t \le \frac{\Delta x^2}{6} . \qquad (6\text{-}8)$$

Let the initial condition for (6-4) be

$$F(x,y,z,0) = F(x,0)\ F(y,0)\ F(z,0) = \left(\tfrac{1}{2} + \text{Erf}\left(\frac{x}{\sigma_{x_0}}\right)\right) \times \left(\tfrac{1}{2} + \text{Erf}\left(\frac{y}{\sigma_{y_0}}\right)\right)\left(\tfrac{1}{2} + \text{Erf}\left(\frac{z}{\sigma_{z_0}}\right)\right) . \qquad (6\text{-}9)$$

In order to select step sizes for (6-6), finite boundaries must be established for the problem. This is done easily since the theoretical solution to (6-4) with initial condition (6-9) is obtained readily [38]. This solution is

$$F(x,y,z,t) = F(x,t)\ F(y,t)\ F(z,t) = \left(\tfrac{1}{2} + \text{Erf}\ \frac{x}{\sigma_x(t)}\right) \times \left(\tfrac{1}{2} + \text{Erf}\left(\frac{y}{\sigma_y(t)}\right)\right)\left(\tfrac{1}{2} + \text{Erf}\left(\frac{z}{\sigma_z(t)}\right)\right) , \qquad (6\text{-}10)$$

where

$$\sigma_x^2(t) = 1 + (\sigma_{x_0}^2 - 1)e^{-2t} , \qquad (6\text{-}11)$$

$$\sigma_y^2(t) = 1 + (\sigma_{y_0}^2 - 1)e^{-2t} ,$$

and

$$\sigma_z^2(t) = 1 + (\sigma_{z_0}^2 - 1)e^{-2t} .$$

For the problem being considered let

$$\sigma_{x_0} = \sigma_{y_0} = \sigma_{z_0} = 0.5 . \qquad (6\text{-}12)$$

Since the solution of (6-4) is the product of three distribution functions which correspond to zero mean Gaussian densities with variances that never exceed unity, boundaries for the problem are easily selected such that all of the probability mass lies within them. Let the boundaries be set such that the solution of (6-4) is sought on

$$-5 \le x \le 5 \,, \tag{6-13}$$

$$-5 \le y \le 5 \,,$$

and

$$-5 \le z \le 5 \,.$$

In accordance with the maximum principle, (6-7), the step sizes selected for the numerical solution are

$$\Delta x = \Delta y = \Delta z = 0.1 \,, \tag{6-14}$$

$$\Delta t = 0.0005 \,.$$

As was the case in previous examples, the modified algorithm is to be operated with P equal to five and using second order polynomial interpolation.

A comparison of the computer time and storage required by the explicit finite difference scheme and the modified algorithm illustrates quite dramatically the substantial reduction in these quantities. The net effect of the reductions in time and storage is to make feasible, numerical solutions which would otherwise be virtually impossible to obtain.

With the selected step sizes, which are fairly large, the explicit finite difference scheme requires

$$(100 + 1)^3 = 1{,}030{,}301$$

spaces of computer storage for F(x,y,z,t). The corresponding amount of storage required by the modified algorithm, with P=5, is only

$$\left(\frac{100}{5} + 1\right)^3 = 9261$$

spaces. This large reduction allows a problem which would have required storage far beyond core storage capability of any computer to fit easily on any large scale general purpose computer. The machine on which this program was executed (Univac 1108) has 192 K core storage, of which a single program is allowed a maximum of 65 K. A portion of this allotted storage must be used to store the computer program in machine language.

Due to the symmetry of this particular problem, it is only necessary to solve (6-4) on

$$- 5 \leq x \leq 0 \ , \qquad (6\text{-}15)$$

$$- 5 \leq y \leq 0 \ ,$$

and

$$- 5 \leq z \leq 0 \ .$$

The solution of (6-4) with initial condition (6-9) was obtained on the range of (x,y,z) given in (6-15). The solution was obtained over two seconds of real time (which is essentially to steady state) in 12 minutes of computer time. A comparable solution using a standard explicit finite difference scheme is estimated to require in excess of 60 minutes of computer time. Because of the large amount of computer time involved, this figure was not checked empirically.

The results of this simulation, which are summarized in Table 6, are very good. The tabulated results are for

Table 6. Errors Obtained in the Solution of a Three-Dimensional Fokker-Planck Equation

Solution at $t =$	Maximum Error $\times 10^{-3}$	Mean Square Error $\times 10^{-6}$
0.5	5.56	6.17
1.0	3.16	2.17
1.5	2.36	1.36
2.0	2.1	1.12

$$F(x,0,0,t) = 0.25\ F(x,t), \ -5 \le x \le 0 \ . \qquad (6\text{-}16)$$

If the modified algorithm is used to solve even higher-dimensional problems, the percentage of savings in computer time and storage over the explicit finite difference scheme becomes even larger.

CHAPTER VII

CONCLUSIONS AND RECOMMENDATIONS

In this research a modified algorithm was developed which efficiently solves parabolic partial differential equations. The modified algorithm is a consistent, convergent, and stable method which greatly reduces the amounts of computer time and storage required to obtain solutions while maintaining the same order of accuracy as the original explicit scheme. The method was tested and its parameters selected by using it to solve the Fokker-Planck equations for several linear systems. The modified algorithm was then used to solve the Fokker-Planck equations for a gated and continuous, first and second order, phase locked loop.

The parameters of the modified algorithm were chosen by solving the Fokker-Planck equations for several linear systems as function of the desired parameters. The most accurate results were obtained from the simulations that used second order polynomial interpolation (see Tables 7, 8, and 9). It was also determined empirically that $P = 5$ (which means that one-fifth of the data in each dimension is stored) should be used.

The selection of P was made by observing the trade-offs between accuracy and the amounts of computer time and storage required for different values of P. The larger the value of P, the larger the savings in computer time and storage. The accuracy of the modified algorithm was observed to remain comparable to that of the original explicit method for values of P less than or equal to five. For larger values of P the accuracy of the method deteriorated quickly (see Tables 7, 8, 9, 10, and 11). It should be noted, however, that not every value of P in the neighborhood of five was investigated and it is possible that a P slightly larger than five (say six or seven) might be a better selection.

The amount of savings in computer time and storage that was realized with the modified algorithm agreed closely with what had been predicted from operation counts. With the parameters selected for the modified algorithm, the solutions obtained for the one-dimensional Fokker-Planck equation for the first order linear system were as accurate as the solutions obtained using the original explicit scheme. The amount of savings realized in computer storage and time for the one-dimensional problem were 80% and about 70% respectively (see Tables 7, 8, and 9). The percent savings in computer storage and time becomes much larger for higher-dimensional problems. For two-dimensional equations the savings in computer storage and time are 96% and about 80% respectively (the savings in computer time was obtained empirically--Table 10).

Even though the Fokker-Planck equation for the coupled second order linear system is not of the exact form for which the maximum principle holds, it was initially used to obtain step sizes for the modified algorithm. Surprisingly, it was observed that the time step size could be much larger than the value suggested by the maximum principle. The algorithm remained stable so long as the time step size was not larger than half the value of the spacial step sizes. It was also observed that accuracy of the algorithm did not strongly depend on the time step size, so long as the method remained stable. The accuracy was primarily a function of the spacial step sizes (see Tables 13, 14, 15, and 16). This is a very helpful result since increasing the size of the time step decreases the amount of computer time required in order to obtain a solution.

The simulation of the three-dimensional Fokker-Planck equation for the third order linear system illustrated clearly that the modified algorithm can solve problems that would otherwise be impractical to solve with the explicit scheme. The solution to this problem was easily obtained with the modified algorithm (see Table 6). A solution to the equation obtained using the original explicit scheme would require a large amount of computer time and an amount of computer storage many times larger than the core capabilities of any large scale general purpose computer.

The modified algorithm was used to obtain complete solutions to the Fokker-Planck equations for a first and second order phase-locked loop. The solutions were started from initial conditions which were uniformly distributed and were run to steady state. Results were obtained for several signal to noise ratios. The numerical and theoretical steady state results for the first order loop agree very closely (see Table 17 and Figures 25, 26, and 27). The steady state solutions for the second order loop agree closely with experimental solutions to a very similar problem presented by Lindsey and Charles [26].

The modified algorithm was also used to solve the Fokker-Planck equations for a first and second order gated phase-locked loop. The objective of these simulations was to find the steady state variances of the phase errors of the systems. Solutions for the first order gated system were obtained for duty factors of 0.1 and 0.05 and for several signal to noise ratios. It was observed that the steady state variances of the phase error for the first order continuous loop and the first order gated loop were the same if both systems received the same amount of signal energy per TDMA time frame (see Figures 41 and 42). Therefore, the steady state variance for the first order gated system can be obtained from a much shorter calculation for the continuous loop or from a simple numerical integration of the theoretical steady state density function for the continuous loop. This could be a useful result for the gated systems with practical duty factors since numerical solutions for such problems would require large amounts of computer time.

The solutions to the Fokker-Planck equations for the second order gated phase-locked loop presented formidable tasks. The relationship which exists between the gated and continuous first order loops was found not hold for the second order systems. Starting from initial conditions which were uniformly distributed, two complete solutions were obtained for the gated second order loop with a duty factor of 0.5, and two partial solutions were obtained for the system with a duty factor of 0.25. These solutions required large amounts of computer time (see Tables 21 and 22) and made it obvious that a complete solution for a second order gated system with a realistic duty factor would be completely impractical to obtain. Such a solution would require an astronomical amount of computer time.

However, the solutions for the second order gated loop also indicate that good estimates of the steady state variances of the phase errors can be made without obtaining complete solutions to the problem. Figures 43 and 44 show that the variances of the phase errors exhibit a decaying oscillatory pattern which make it possible to predict the steady state variances at early stages of the solutions.

There are many interesting topics related to those which were studied in this thesis which deserve further consideration. It is believed that the polynomial technique which was used to modify the explicit finite difference scheme can also be applied beneficially to other basic methods. It is clear that this is the case for implicit finite difference schemes.

The Fokker-Planck equation is a general method of analyzing systems with random disturbances. This equation has many interesting and important applications, and solutions to it are sought in many areas. It is hoped that the work in this thesis might be helpful in some other fields.

There is much additional important information which can be obtained for phase-locked loops. One such quantity is the solution to the Fokker-Planck for systems when it is assumed that the frequency of the transmitted signal is not known exactly ($\omega_o \neq \omega$ in (5-2) and (5-14)). Another interesting problem is to consider the statistics of the phase errors of loops for different types of received signals (an f.m. signal for instance). An important and very difficult problem is the solution of Fokker-Planck equations for higher order phase-locked loops (third order in particular).

Many of the important calculations for phase-locked loops require large amounts of computer time. Therefore, it would be useful to perform some parametric studies in order to try and establish some rules for approximating desired information. One area where this could be very helpful is in the study of the frequency of skipping of cycles for a system. Another helpful area would be determining how long it takes the variance of the phase error of a system, for a given initial density function, to fall, and remain, below a certain level.

APPENDICES

APPENDIX A

This appendix contains the details of the numerical solutions of the Fokker-Planck equations for the linear systems. The data contained herein is from the solutions of equations (4-14), (4-25), and (4-30).

Table 7. Results for One-Dimensional Linear System--$\Delta x = 0.1$, $\Delta t = 0.005$, $\sigma_0^2 = 1.0$, $t = 0.025$

Degree of Approx. Polynomial	Storage	Computer Time (ms)	Estimated Savings in Computer Time (Low Estimate) (%)	Savings in Computer Time (%)	Maximum Error $\times 10^{-3}$	Mean Square Error $\times 10^{-10}$
Classical Method	N	450	----	----	0.57	167
3	N/5	155	57	66	1.69	5350
2	N/5	158	68	65	0.32	405
3	N/4	175	46	61	1.0	1704
2	N/4	159	59	65	0.21	160
3	N/3	219	28	51	0.41	345
2	N/3	165	46	63	0.11	53

Table 8. Results for the One-Dimensional Linear System--$\Delta x = 0.01$, $\Delta t = 0.005$, $\sigma_0^2 = 0.25$, $t = 0.5$

Degree of Approx. Polynomial	Storage	Computer Time (ms)	Estimated Savings in Computer Time (Low Estimate) (%)	Savings in Computer Time (%)	Maximum Error $\times 10^{-2}$	Mean Square Error $\times 10^{-4}$
Classical Method	N	419	----	----	1.08	0.3079
3	N/5	151	57	64	3.59	4.25
2	N/5	186	68	56	0.28	0.0170
3	N/4	188	46	55	2.52	1.99
2	N/4	175	59	58	0.51	0.0750
3	N/3	204	28	51	1.67	0.7982
2	N/3	218	46	48	0.84	0.1698

Table 9. Results for the One-Dimensional Linear System--$\Delta x = 0.01$, $\Delta t = 0.005$, $\sigma_0^2 = 4.0$, $t = 0.5$

Degree of Approx. Polynomial	Storage	Computer Time (ms)	Estimated Savings in Computer Time (Low Estimate) (%)	Savings in Computer (%)	Maximum Error $\times 10^{-2}$	Mean Square Error $\times 10^{-4}$
Classical Method	N	996	----	----	1.53	0.472298
3	N/5	342	57	66	1.83	0.94503
2	N/5	279	68	72	1.37	0.52648
3	N/4	391	46	61	1.66	0.754153
2	N/4	331	59	67	1.41	0.497814
3	N/3	640	28	36	1.57	0.685472
2	N/3	426	46	57	1.49	0.522899

Table 10. Results for the Simple Second Order Linear System-- $\Delta x_1 = \Delta x_2 = 0.05$, $\Delta t = 0.0005$, $\sigma^2_{1_0} = \sigma^2_{2_0} = 1.0$, $t = 0.0025$

Storage	Solution Obtained on $\le (x_1 \text{ and } x_2) \le$	Computer Time Required (ms)	Maximum Error $\times 10^{-5}$	Mean Square Error $\times 10^{-10}$
N (explicit method)	[-5,0]	6972	6.0	2.02
N/9	[-5.1,0]	2445	1.0	0.0
N/16	[-6,0]	2159	0.0	0.0
N/25	[-5,0]	1467	1.0	0.12
N/100	[-5,0]	793	1.0	0.21
N/225	[-5.25,0]	552	4.0	1.45
N/400	[-5,0]	490	5.0	4.4
N/625	[-5,0]	298	6.0	5.63

Table 11. Results for the Simple Second Order Linear System-- $\Delta x_1 = \Delta x_2 = 0.05$, $\Delta t = 0.0005$, $\sigma^2_{1_0} = \sigma^2_{2_0} = 0.25$, $t = 0.5$

Amount of Storage	Solution Obtained on $\le (x_1 \text{ and } x_2) \le$	Computer Time Required (ms)	Maximum Error $\times 10^{-3}$	Mean Square Error $\times 10^{-5}$
N (explicit method)				
N/25	[-5,0]	24780	0.76	0.006
N/100	[-5,0]	7245	3.60	0.074
N/400	[-7,0]	3716	15.69	2.23

Table 12. Results Obtained for the Coupled Second Order System--$\Delta x = \Delta z = 0.04$, $-5 \le x \le 5$, $-5 \le z \le 5$, $\sigma^2_{x_0} = \sigma^2_{z_0} = 0.25$, $t = 0.8$

Time Step $\Delta t =$	Computer Time Required (ms)	Maximum Error $\times 10^{-4}$	Mean Square Error $\times 10^{-10}$
0.0004	127634	0.6	6.35
0.0008	59307	0.6	5.75
0.001	49597	0.6	5.2
0.0016	32019	0.6	5.6
0.002	24440	0.6	6.1

Table 13. Stability and Accuracy Study as a Function of Δt for the Coupled Second Order System--$\Delta x = \Delta z = 0.04$, $-5 \le x \le 5$, $-5 \le z \le 5$, $\sigma^2_{x_0} = \sigma^2_{z_0} = 0.25$

Time Steps $\Delta t =$	Solution at $t =$	Computer Time Required (ms)	Maximum Error $\times 10^{-3}$	Mean Square Error $\times 10^{-10}$
0.0008	0.2	152293	1.29	866
0.01	0.2	12301	0.67	218
0.01	1.0	61526	2.32	6631
0.02	1.0	28130	2.4	6127
0.04	1.0	---	U N S T	A B L E

Table 14. Accuracy Study as a Function of Δt for the Coupled Second Order System--$\Delta x = \Delta z = 0.1$, $-6 \leq x \leq 6$, $-6 \leq z \leq 6$, $\sigma^2_{x_0} = \sigma^2_{z_0} = 0.25$

Time Step $\Delta t =$	Solution at $t =$	Computer Time Required (ms)	Maximum Error $\times 10^{-3}$	Mean Square Error $\times 10^{-5}$
0.002	1.0	66170	7.52	0.85
0.02	1.0	7270	7.68	0.84
0.05	1.0	3023	8.07	0.82
0.02	2.5	18165	11.67	1.74
0.05	2.5	7555	10.40	1.64
0.02	4.0	29081	12.63	2.58
0.05	4.0	12101	177.25	365.00

Table 15. Accuracy Study as a Function of Δx, Δz, and Δt for the Coupled Second Order System--$\sigma^2_{x_0} = \sigma^2_{z_0} = 0.25$

Space Steps $\Delta x = \Delta z =$	Time Step $\Delta t =$	Solution at $t =$	Solved on $[x_\ell, x_u] \times [z_\ell, z_u]$*	Computer Time Required (ms)	Maximum Error $\times 10^{-3}$	Mean Square Error $\times 10^{-10}$
0.04	0.01	0.2	$[-5,5] \times [-5,5]$	12301	0.67	218
0.1	0.002	0.2	$[-6,6] \times [-6,6]$	13224	6.30	411
0.04	0.01	0.6	$[-5,5] \times [-5,5]$	36916	1.41	2099
0.1	0.002	0.6	$[-6,6] \times [-6,6]$	39702	8.29	63800
0.04	0.01	1.0	$[-5,5] \times [-5,5]$	61526	2.32	6632
0.1	0.002	1.0	$[-6,6] \times [-6,6]$	66170	7.52	85300

*The ℓ and u subscripts indicate the lower and upper bounds on the respective variables.

Table 16. Accuracy Study as a Function of Δx and Δz for the Coupled Second Order System--$\Delta t = 0.02$, $\sigma^2_{x_0} = \sigma^2_{z_0} = 0.25$

Space Steps $\Delta x = \Delta z =$	Solution at $t =$	Solved on $[x_\ell, x_u]\times[z_\ell, z_u]$*	Computer Time Required (ms)	Maximum Error $\times 10^{-3}$	Mean Square Error $\times 10^{-5}$
0.04	1.0	$[-5,5]\times[-5,5]$	28130	2.40	0.06
0.1	1.0	$[-6,6]\times[-6,6]$	7270	7.68	0.84
0.04	2.5	$[-5,5]\times[-5,5]$	70325	1.19	0.02
0.1	2.5	$[-6,6]\times[-6,6]$	18165	11.67	0.17
0.04	4.0	$[-5,5]\times[-5,5]$	112520	2.46	0.09
0.1	4.0	$[-6,6]\times[-6,6]$	29081	12.63	2.58

*The ℓ and u subscripts indicate the lower and upper bounds on the respective variables.

APPENDIX B

This appendix contains the details of the numerical solutions of the Fokker-Planck equations for the phase-locked loops. The data contained herein is from the solutions of equations (5-11), (5-41), (5-43), and (5-44).

Table 17. Results of the Fokker-Planck Equation for the Continuous First Order Phase-Locked Loop--$\Delta\phi = \pi/40$, $\Delta\tau = 0.003$

SNR $\alpha =$	Solution Obtained to $\tau =$ sec.	Computer Time Required (ms)	Steady State Reached at App. $\tau =$ sec.	Maximum Steady State Error $\times 10^{-3}$	Mean Square Steady State Error $\times 10^{-5}$
0.5	15	---	0.9	1.25	0.065
1	30	10034	2.0	3.2	0.392
2	15	5223	4.0	8.83	2.234

Table 18. Results of the Fokker-Planck Equation for the Gated First Order Phase-Locked Loop--$\Delta\phi = \pi/40$, $\Delta\tau = 0.001$, Time Frame = 0.5, Duty Factor = 0.1

SNR α =	Number of Cycles Obtained	Computer Time Required (sec.)	Number of Cycles Required to Reach Steady State (Approximate)
2	35	9.8	30
5	100	27.8	60
10	100	28.1	100
20	150	45.5	145
30	150	48.6	150
40	150	47.3	150

Table 19. Results of the Fokker-Planck Equation for the Gated First Order Phase-Locked Loop--$\Delta\phi = \pi/40$, $\Delta\tau = 0.001$, Time Frame = 0.5, Duty Factor = 0.05

SNR $\alpha =$	Number of Cycles Obtained	Computer Time Required (sec.)	Number of Cycles Required to Reach Steady State (Approximate)
10	150	46.7	130
20	250	66.5	230
40	300	86.7	300

Table 20. Results of the Fokker-Planck Equation for the Continuous Second Order Phase-Locked Loop--$\Delta\phi = \pi/50$.

SNR σ =	$\Delta\tau$	Solution Obtained to τ = sec.	Computer Time Required (min.)	Approx. Time To Reach Steady State τ = sec.	Steady State Value of Variance
0.41 (-3.87 db)	0.005	8.0	3.24	2.5	---
1.1 (0.41 db)	0.001	12.0	22.18	3.8	2.09
1.382 (1.41 db)	0.001	15.0	27.82	4.2	1.73
2.76 (4.41 db)	0.001	8.0	16.46	8.0	---

Table 21. Results of the Fokker-Planck Equation for the Gated Second Order Phase-Locked Loop--$\Delta\phi = \pi/50$, $\Delta\tau = 0.001$, Time Frame = 0.5, Duty Factor = 0.5

SNR α =	Number of Cycles Obtained	Computer Time Required (min.)	Number of Cycles Required to Reach Steady State (Approximate)
2.2	30	25.47	27
4.0	35	30.33	31

Table 22. Results of the Fokker-Planck Equation for the Gated Second Order Phase-Locked Loop--$\Delta\phi = \pi/50$, $\Delta\tau = 0.001$, Time Frame = 0.5, Duty Factor = 0.25

SNR $\alpha =$	Number of Cycles Obtained	Computer Time Required (min.)
6.0	45	35.56
9.0	20	16.77

APPENDIX C

This appendix contains a representative selection of the computer programs used to obtain the numerical results for the phase-locked loops. While not showing every program that was used, the four which are given do generate all the information presented in Chapter V. Each of these programs uses the modified algorithm to obtain its results.

The first program generates the density function, distribution function, and the variance of the phase error for the continuous (not gated) first order phase-locked loop.

The second program generates the variance of the phase error for the gated first order phase-locked loop.

The third program generates the density function, distribution function, and variance of the phase error for the continuous (not gated) second order phase-locked loop.

The fourth program generates the variance of the phase error for the gated second order phase-locked loop.

```
C*****************************************************************
C
C
C           FIRST ORDER PHASE-LOCKED LOOP
C           SNR=2
C           SECOND ORDER POLYNOMIAL INTERPOLATION
C           ONE FIFTH STORAGE
C
C
C*****************************************************************
      DIMENSION ANSW(15,15),ANS(10),NANS(10),POLY(5),D(15,15)
     X,VAR(15)
       REAL NANS
1     FORMAT(1H0,1X,3HANG,7X,3H.15,7X,3H0.3,7X,3H.45,7X,3H0.6,
     X7X,3H.75,7X,3H0.9,7X,4H1.05,6X,4H1.20,6X,4H1.35,6X,4H1.50)
2     FORMAT(1H ,1X,I1,5X,11(F10.5))
3     FORMAT()
4     FORMAT(1H ,7X,11(F10.5))
20    FORMAT(1H ,50H**************************************************)
21    FORMAT(1H0,1H )
22    FORMAT(1H ,29HFIRST ORDER PHASE-LOCKED LOOP)
23    FORMAT(1H ,5HSNR=2)
24    FORMAT(1H ,37HSECOND ORDER POLYNOMIAL APPROXIMATION)
25    FORMAT(1H ,17HONE FIFTH STORAGE)
26    FORMAT(1H+,6H- PI/8)
27    FORMAT(1H0,21HDISTRIBUTION FUNCTION)
28    FORMAT(1H0,16HDENSITY FUNCTION)
29    FORMAT(1H0,18HVARIANCE  E(X**2)=)
      DELX=0.07853982
      DELT=0.003
      PS=5.0
      SNR=2.0
      DO 70 I=1,8
70    ANS(I)=0.3926991*0.1591549*(I-1)
      ANS(9)=0.5
      ANS(1)=0.0
      PSTP=PS*DELX
      A=DELT/(2.0*DELX)
      B=DELT/(SNR*DELX**2)
      C=1-2.0*B
      E=1.0/(2.0*DELX)
      KK=(3.1415927+PSTP)/PSTP+0.5
      K=KK-2
      NANS(KK)=0.5
      DO 10 I=1,KK
10    ANSW(1,I)=ANS(I)
      VAR(1)=3.1415927**2/3.0
      DO 7) KKK=2,11
      DO 11 J=1,50
      DO 13 I=1,K
      FO=ANS(I)
      DFO=ANS(I+1)-ANS(I)
      D2FO=ANS(I+2)-ANS(I+1)-DFO
      POLY(1)=FO+0.8*DFO-0.08*D2FO
```

```
       IF(POLY(1)-ANS(I))60,61,61
 60    POLY(1)=ANS(I)
 61    CONTINUE
       POLY(2)=F0+1.2*DF0+0.12*D2F0
       X=-3.1415927+I*PSTP
 13    NANS(I+1)=POLY(2)*(A*SIN(X)+B)+ANS(I+1)*C+POLY(1)*(B-A*SIN(X))
       DO 51 I=1,KK
 51    ANS(I)=NANS(I)
 11    CONTINUE
       VA=0.0
       DO 100 IT=1,K,2
 100   VA=VA+ANS(IT)*(3.1415927-(IT-1)*PSTP)+4.0*ANS(IT+1)*(
      X3.1415927-IT*PSTP)+ANS(IT+2)*(3.1415927-(IT+1)*PSTP)
       VAR(KKK)=VA*PSTP*4.0/3.0
       DO 72 I=1,KK
 72    ANSW(KKK,I)=ANS(I)
       POLY(2)=0.12*ANS(2)
       D(KKK,1)=POLY(2)
       DO 75 I=1,K
       F0=ANS(I)
       DF0=ANS(I+1)-ANS(I)
       D2F0=ANS(I+2)-ANS(I+1)-DF0
       POLY(1)=F0+0.8*DF0-0.08*D2F0
       IF(POLY(1)-ANS(I))80,81,81
 80    POLY(1)=ANS(I)
 81    CONTINUE
       POLY(2)=F0+1.2*DF0+0.12*D2F0
 75    D(KKK,I+1)=(POLY(2)-POLY(1))*E
       F0=ANS(KK-1)
       DF0=ANS(KK)-ANS(KK-1)
       POLY(1)=F0+0.8*DF0
       POLY(2)=F0+1.2*DF0
       D(KKK,KK)=(POLY(2)-POLY(1))*E
 71    CONTINUE
```

```
      WRITE(6,20)
      WRITE(6,21)
      WRITE(6,22)
      WRITE(6,23)
      WRITE(6,24)
      WRITE(6,25)
      WRITE(6,21)
      WRITE(6,20)
      WRITE(6,21)
      WRITE(6,27)
      WRITE(6,1)
      DO 17 I=1,KK
      M=9-I
      WRITE(6,2)M,ANSW(1,I),ANSW(2,I),ANSW(3,I),ANSW(4,I),ANSW(5,I),
     XANSW(6,I),ANSW(7,I),ANSW(8,I),ANSW(9,I),ANSW(10,I),ANSW(11,I)
   17 WRITE(6,26)
      WRITE(6,28)
      WRITE(6,21)
      DO 85 I=1,KK
      M=9-I
      WRITE(6,2)M,D(1,I),D(2,I),D(3,I),D(4,I),D(5,I),
     XD(6,I),D(7,I),D(8,I),D(9,I),D(10,I),D(11,I)
   85 WRITE(6,26)
      WRITE(6,29)
      WRITE(6,21)
      WRITE(6,4)VAR(1),VAR(2),VAR(3),VAR(4),VAR(5),VAR(6),VAR(7),
     XVAR(8),VAR(9),VAR(10),VAR(11)
      END
```

```
C*****************************************************************
C
C
C          FIRST ORDER PHASE-LOCKED LOOP
C          SNR=0
C          SECOND ORDER POLYNOMIAL INTERPOLATION
C          ONE FIFTH STORAGE
C
C
C*****************************************************************
      DIMENSION ANS(10),NANS(10),POLY(5),VAR(400,35)
       REAL NANS
1     FORMAT(1H ,1X,3HANG,7X,3H.15,7X,3H0.3,7X,3H.45,7X,3H0.6,
     X7X,3H.75,7X,3H0.9,7X,4H1.05,6X,4H1.20,6X,4H1.35,6X,4H1.50)
2     FORMAT(1H ,1X,I1,5X,11(F10.5))
3     FORMAT()
4     FORMAT(1H ,7X,11(F10.5))
5     FORMAT(1H ,2X,4HTIME,5(7X,5HCYCLE))
6     FORMAT(1H ,1X,F6.4,5(5X,F7.4))
7     FORMAT(1H ,5X,5(9X,I3))
15    FORMAT(1H ,9HVARIANCES)
20    FORMAT(1H ,50H**************************************************)
21    FORMAT(1H ,1H )
22    FORMAT(1H ,29HFIRST ORDER PHASE-LOCKED LOOP)
23    FORMAT(1H ,6HSNR=40)
24    FORMAT(1H ,37HSECOND ORDER POLYNOMIAL APPROXIMATION)
25    FORMAT(1H ,17HONE FIFTH STORAGE)
26    FORMAT(1H ,6H= PI/8)
27    FORMAT(1H ,21HDISTRIBUTION FUNCTION)
```

```
28    FORMAT(1H0,16HDENSITY FUNCTION)
29    FORMAT(1H ,18HVARIANCE  E(X**2)=)
      DELX=0.0747539A2
      DELT=0.001
      PS=5.0
      SHR=40.0
      DO 70 I=1,8
70    ANS(I)=0.3926991*0.1591549*(I-1)
      ANS(9)=0.5
      ANS(1)=0.0
      PSTP=PS*DELX
      A=DELT/(2.0*DELX)
      B=DELT/(SHR*DELX**2)
      C=1-2.0*B
      E=1.0/(2.0*DELX)
      KK=(3.1415927+PSTP)/PSTP+0.5
      K=KK-2
      NANS(KK)=0.5
      VAR(1,1)=3.1415927**2/3.0
      DO 30 N=1,300
      VAR(N,1)=VAR(N-1,21)
      DO 31 J=2,2
      DO 32 JJ=1,25
      DO 33 I=1,K
      F0=ANS(I)
      DFO=ANS(I+1)-ANS(I)
      D2FO=ANS(I+2)-ANS(I+1)-DFO
      POLY(1)=F0+0.8*DFO-0.08*D2FO
      IF(POLY(1)-ANS(I))60,61,61
60    POLY(1)=ANS(I)
61    CONTINUE
      POLY(2)=F0+1.2*DFO+0.12*D2FO
      X=-3.1415927+I*PSTP
33    NANS(I+1)=POLY(2)*(A*SIN(X)+B)+ANS(I+1)*C+POLY(1)*(B-A*SIN(X))
      DO 51 I=1,KK
51    ANS(I)=NANS(I)
32    CONTINUE
      VA=0.0
      DO 100 IT=1,K,2
100   VA=VA+ANS(IT)*(3.1415927-(IT-1)*PSTP)+4.0*ANS(IT+1)*(
     X3.1415927-IT*PSTP)+ANS(IT+2)*(3.1415927-(IT+1)*PSTP)
      VAR(N,J)=VA*PSTP*4.0/3.0
31    CONTINUE
      DO 34 J=3,21
      DO 35 JJ=1,25
      DO 36 I=1,K
      F0=ANS(I)
      DFO=ANS(I+1)-ANS(I)
      D2FO=ANS(I+2)-ANS(I+1)-DFO
      POLY(1)=F0+0.8*DFO-0.08*D2FO
      IF(POLY(1)-ANS(I))80,81,81
80    POLY(1)=ANS(I)
81    CONTINUE
      POLY(2)=F0+1.2*DFO+0.12*D2FO
      X=-3.1415927+I*PSTP
36    NANS(I+1)=POLY(2)*B+ANS(I+1)*C+POLY(1)*B
      DO 52 I=1,KK
52    ANS(I)=NANS(I)
35    CONTINUE
      VA=0.0
      DO 101 IT=1,K,2
101   VA=VA+ANS(IT)*(3.1415927-(IT-1)*PSTP)+4.0*ANS(IT+1)*(
     X3.1415927-IT*PSTP)+ANS(IT+2)*(3.1415927-(IT+1)*PSTP)
      VAR(N,J)=VA*PSTP*4.0/3.0
34    CONTINUE
30    CONTINUE
      WRITE(6,21)
      WRITE(6,21)
      WRITE(6,25)
      WRITE(6,27)
      WRITE(6,25)
      WRITE(6,25)
      WRITE(6,21)
      WRITE(6,21)
      WRITE(6,21)
      VAR(1,1)=3.1415927**2/3.0
      DO 110 N=1,300,5
      N1=N
      N2=N+1
      N3=N+2
      N4=N+3
      WRITE(6,21)
      WRITE(6,21)
      N5=N+4
      WRITE(6,5)
      WRITE(6,7)N1,N2,N3,N4,N5
      DO 111 J=1,21
      T=(J-1)*0.025
111    WRITE(6,6)T,VAR(N1,J),VAR(N2,J),VAR(N3,J),VAR(N4,J),VAR(N5,J)
110   CONTINUE
      END
```

```
C*************************************************************************
C
C
C          SECOND ORDER PHASE LOCK LOOP
C          SNR=1.382
C          INTEGRATER GAIN=AK
C          SECOND ORDER POLYNOMIAL APPROXIMATION
C          DELY=DELZ=PI/50        DELT=0.001
C          ONE FIFTH STORAGE
C
C
C*************************************************************************
      DIMENSION ANS(21,21),NANS(21,21),AIX(21,21),AIZ(21,21),POLX(2),
     XPOLZ(2),POXZ(2),POZX(2),DX(25),DZ(25),VAR(15),TIM(15),DI(25,15),
     XDEL(25,15)
      REAL NANS
1     FORMAT(1H ,11X,5HX=-PI,4X,7HX=-.8PI,3X,7HX=-.6PI,3X,7HX=-.4PI,3X,
     X7HX=-.2PI,3X,3HX=0,7X,6HX=.2PI,4X,6HX=.4PI,4X,6HX=.6PI,4X,
     X6HX=.8PI,4X,4HX=PI)
2     FORMAT(1H ,I2,5X,11(F10.5))
3     FORMAT()
4     FORMAT(1H ,5HTIME=)
5     FORMAT(1H+,5X,F6.4)
6     FORMAT(1H ,4X,1HX,6X,2HT=,6(8X,2HT=))
7     FORMAT(1H+,8X,7(5X,F5.2))
8     FORMAT(1H ,I2,6X,7(F10.5))
9     FORMAT(1H+,3X,5HP1710)
10    FORMAT(1H0,17HVARIANCE F(X**2)=)
11    FORMAT(1H0,8X,7(2X,F8.5))
20    FORMAT(1H ,75H**********************************************************
     X*************************)
21    FORMAT(1H0,1H )
22    FORMAT(1H ,28HSECOND ORDER PHASE LOCK LOOP)
23    FORMAT(1H ,9HSNR=1.382)
24    FORMAT(1H ,18HINTEGRATER GAIN=AK)
```

```
25    FORMAT(1H ,37HSECOND ORDER POLYNOMIAL APPROXIMATION)
26    FORMAT(1H ,17HONE FIFTH STORAGE)
27    FORMAT(1H ,21HDISTRIBUTION FUNCTION)
28    FORMAT(1H ,16HDENSITY FUNCTION)
29    FORMAT(1H+,2X,4HPI/5)
30    FORMAT(1H1,1H )
31    FORMAT(1H ,32HDELX=DELZ=PI/50       DELT=0.005 )
32    FORMAT(1H ,10X,1HX,10X,4HF(X),15X,1HZ,10X,4HF(Z))
33    FORMAT(1H ,6X,I3,10X,F10.5,7X,I3,10X,F10.5)
34    FORMAT(1H+,9X,5HPI/10,25X,5HPI/10)
35    FORMAT(1H ,I10)
36    FORMAT(1H ,28HANS WAS NOT COPIED CORRECTLY)
37    FORMAT(1H ,28HAIX WAS NOT COPIED CORRECTLY)
38    FORMAT(1H ,28HAIZ WAS NOT COPIED CORRECTLY)
39    FORMAT(1H ,23HNUMBER OF WORDS STORED=)
42    FORMAT(1H+,25X,I10)
      DELX=0.062831853
      DELZ=0.062831853
      DELT=0.001

      PS=5.0
      SNR=1.382
      IS=1
      TIM(1)=0.0
      TIM(2)=0.5
      TIM(3)=1.0
      TIM(4)=1.5
      TIM(5)=2.0
      TIM(6)=2.5
      TIM(7)=3.0
      PSX=PS*DELX
      PSZ=PS*DELZ
      II=(6.2831853+PSX)/PSX+0.5
      JJ=(6.2831853+PSZ)/PSZ+0.5
      III=II-1
      JJJ=JJ-1
      IM2=TI-2
      TI=TIM(I)
      A=DELT/(2*DELX)
      B=DELT/(2*SNR*DELX**2)
      C=DELT/(2*DELZ)
      D=1.0-2*B
      E=1.0-B
      F=1/(2*DELX)
      G=1/(2*DELZ)
      T=TI
      CALL NTRAN(16,2,441 ,ANS,L)
      CALL NTRAN(16,22)
      IF(L.NE.441 )GO TO 90
      GO TO 91
90    WRITE(6,36)
      WRITE(6,35)L
      GO TO 99
91    CALL NTRAN(17,2,441 ,AIX,L)
      CALL NTRAN(17,22)
      IF(L.NE.441 )GO TO 92
      GO TO 93
92    WRITE(6,37)
      WRITE(6,35)L
      GO TO 99
93    CALL NTRAN(18,2,441 ,AIZ,L)
      CALL NTRAN(18,22)
      IF(L.NE.441 )GO TO 94
      GO TO 95
94    WRITE(6,38)
      WRITE(6,35)L
      GO TO 99
95    CALL NTRAN(19,2,1,VA,L)
      CALL NTRAN(19,22)
      IF(L.NE.1) GO TO 96
      GO TO 97
96    WRITE(6,36)
      WRITE(6,35)L
      GO TO 99
97    CONTINUE
      DO 70 I=1,21
70    D1(I,1)=ANS(I,JJ)

      VAR(1)=VA
      DX(1)=0.12*ANS(2,JJ)*F
      DO 102 K=2,20
      I=1+(K-1)*I5
      FO =ANS(I-1,JJ)
      DFO=ANS(I,JJ)-ANS(I-1,JJ)
      D2FO=ANS(I+1,JJ)-ANS(I,JJ)-DFO
      POLX(1)=FO+0.8*DFO-0.08*D2FO
      POLX(2)=FO+1.2*DFO+0.12*D2FO
102   DX(K)=(POLX(2)-POLX(1))*F
      FO=ANS(II-1,JJ)
      DFO=1.0-ANS(II-1,JJ)
      D2FO=-DFO
      POLX(1)=FO +0.8*DFO-0.08*D2FO
      DX(21)=(1.0-POLX(1))*F
      DO 213 I=1,21
213   DL(I,1)=DX(I)
      DO 80 LL=2,7
      L2=(TIM(LL)-TIM(LL-1))/DELT+0.5
      T=TI+DELT*(LL-1)*L2
      DO 75 L=1,L2
      DO 40 J=2,JJJ
      Z=-3.1415927+(J-1)*PSZ
      DO 41 I=2,III
      X=-3.1415927+(I-1)*PSX
      FOX=ANS(I-1,J)
      DFOX=ANS(I,J)-ANS(I-1,J)
      D2FOX=ANS(I+1,J)-ANS(I,J)-DFOX
      FOZ=ANS(I,J-1)
      DFOZ=ANS(I,J)-ANS(I,J-1)
      D2FOZ=ANS(I,J+1)-ANS(I,J)-DFOZ
      DXZ=AIZ(I-1,J)
      DDXZ=AIZ(I,J)-AIZ(I-1,J)
      D2DXZ=AIZ(I+1,J)-AIZ(I,J)-DDXZ
      DZX=AIX(I,J-1)
      DDZX=AIX(I,J)-AIX(I,J-1)
      D2DZX=AIX(I,J+1)-AIX(I,J)-DDZX
      POLX(1)=FOX+0.8*DFOX-0.08*D2FOX
      POLX(2)=FOX+1.2*DFOX+0.12*D2FOX
      POLZ(1)=FOZ+0.8*DFOZ-0.08*D2FOZ
      POLZ(2)=FOZ+1.2*DFOZ+0.12*D2FOZ
      PDXZ(1)=DXZ+0.8*DDXZ-0.08*D2DXZ
      PDXZ(2)=DXZ+1.2*DDXZ+0.12*D2DXZ
      PDZX(1)=DZX+0.8*DDZX-0.08*D2DZX
      PDZX(2)=DZX+1.2*DDZX+0.12*D2DZX
      NANS(I,J)=POLX(2)*(Z*A-X*A+SIN(X)*A+B)+POLX(1)*(-Z*A+X*A-SIN(X)*A
     X+D)+POLZ(2)*(Z*C-X*C)+POLZ(1)*(-Z*C+X*C)+ANS(I,J)*D+PDZX(2)*C
     X-PDZX(1)*C-PDXZ(2)*A+PDXZ(1)*A
41    CONTINUE
      I=II
      X=-3.1415927+(I-1)*PSX
      FOZ=ANS(I,J-1)
      DFOZ=ANS(I,J)-ANS(I,J-1)
      D2FOZ=ANS(I,J+1)-ANS(I,J)-DFOZ
      DZX=AIX(I,J-1)
      DDZX=AIX(I,J)-AIX(I,J-1)
      D2DZX=AIX(I,J+1)-AIX(I,J)-DDZX
      POLZ(1)=FOZ+0.8*DFOZ-0.08*D2FOZ
      POLZ(2)=FOZ+1.2*DFOZ+0.12*D2FOZ
      PDZX(1)=DZX+0.8*DDZX-0.08*D2DZX
      PDZX(2)=DZX+1.2*DDZX+0.12*D2DZX
      NANS(I,J)=ANS(I,J)+POLZ(2)*(Z*C-X*C)+POLZ(1)*(-Z*C+X*C)+PDZX(2)*C
     X-PDZX(1)*C
40    CONTINUE
      J=JJ
      Z=-3.1415927+(J-1)*PSZ
      DO 65 I=2,III
      X=-3.1415927+(I-1)*PSX
      FOX=ANS(I-1,J)
      DFOX=ANS(I,J)-ANS(I-1,J)
      D2FOX=ANS(I+1,J)-ANS(I,J)-DFOX
      DXZ=AIZ(I-1,J)
      DDXZ=AIZ(I,J)-AIZ(I-1,J)
      D2DXZ=AIZ(I+1,J)-AIZ(I,J)-DDXZ
      POLX(1)=FOX+0.8*DFOX-0.08*D2FOX
      POLX(2)=FOX+1.2*DFOX+0.12*D2FOX
      PDXZ(1)=DXZ+0.8*DDXZ-0.08*D2DXZ
      PDXZ(2)=DXZ+1.2*DDXZ+0.12*D2DXZ
65    NANS(I,J)=POLX(2)*(Z*A-X*A+SIN(X)*A+B)+POLX(1)*(-Z*A+X*A-SIN(X)*A
     X+B)+ANS(I,J)*D-PDXZ(2)*A+PDXZ(1)*A
      NANS(II,JJ)=1.0
      DO 48 J=2,JJ
      DO 48 I=2,II
48    ANS(I,J)=NANS(I,J)
      DO 61 J=1,JJ
      AIX(1,J)=0.0
      AIX(2,J)=ANS(2,J)*PSX*0.5
61    AIX(3,J)=(ANS(3,J)+4.0*ANS(2,J))*PSX*0.3333333
      DO 49 I=4,III,2
      DO 49 J=1,JJ
      AIX(I,J)=AIX(I-2,J)+(ANS(I,J)+4.0*ANS(I-1,J)+ANS(I-2,J))*PSX*0.333
     X3333
49    AIX(I+1,J)=AIX(I-1,J)+(ANS(I+1,J)+4.0*ANS(I,J)+ANS(I-1,J))*PSX*0.3
     X333333
      DO 62 I=1,II
      AIZ(I,1)=0.0
      AIZ(I,2)=ANS(I,2)*PSZ*0.5
62    AIZ(I,3)=(ANS(I,3)+4.0*ANS(I,2))*PSZ*0.3333333
      DO 50 J=4,JJJ,2
      DO 50 I=1,II
      AIZ(I,J)=AIZ(I,J-2)+(ANS(I,J)+4.0*ANS(I,J-1)+ANS(I,J-2))*PSZ*0.333
     X3333
50    AIZ(I,J+1)=AIZ(I,J-1)+(ANS(I,J+1)+4.0*ANS(I,J)+ANS(I,J-1))*PSZ*0.3
     X333333
75    CONTINUE
      DX(1)=0.12*ANS(2,JJ)*F
      DO 72 K=2,20
      I=1+(K-1)*IS
      F0 =ANS(I-1,JJ)
      DF0=ANS(I,JJ)-ANS(I-1,JJ)
      D2F0=ANS(I+1,JJ)-ANS(I,JJ)-DF0
      POLX(1)=F0+0.8*DF0-0.08*D2F0
      POLX(2)=F0+1.2*DF0+0.12*D2F0
72    DX(K)=(POLX(2)-POLX(1))*F
```

```
      FO=ANS(II-1,JJ)
      DFO=1.0-ANS(II-1,JJ)
      D2FO=-DFO
      POLX(1)=FO +0.6*DFO-0.08*D2FO
      PX(21)=(1.0-POLX(1))*G
      DZ(1)=0.72*ANS(II,2)*G
      DO 75 K=2,20
      J=1+(K-1)*15
      FO=ANS(II,J-1)
      DFO=ANS(II,J)-ANS(II,J-1)
      D2FO=ANS(II,J+1)-ANS(II,J)-DFO
      POLZ(1)=FO+0.8*DFO-0.08*D2FO
      POLZ(2)=FO+1.2*DFO+0.12*D2FO
75    DZ(K)=(POLZ(2)-POLZ(1))*G
      FO=ANS(II,JJ-1)
      DFO=1.0-ANS(II,JJ-1)
      D2FO=-DFO
      POLZ(1)=FO+0.8*DFO-0.08*D2FO
      PZ(21)=(1.0-POLZ(1))*G
      DO 210 I=1,21
      DA(I,LL)=ANS(I,JJ)
210   DL(I,LL)=DX(I)
      VA=0.0
      DO 200 I=1,IM2,2
200   VA=VA+ANS(I,JJ)*(3.1415927-(I-1)*PSX)+4.0*ANS(I+1,JJ)*
     X(3.1415927-I*PSX)+ANS(I+2,JJ)*(3.1415927-(I+1)*PSX)
      VAR(LL)=3.1415927**2+2.0*VA*PSX/3.0
      VA=VAR(LL)
80    CONTINUE
```

```
      WRITE(6,20)
      WRITE(6,21)
      WRITE(6,22)
      WRITE(6,23)
      WRITE(6,24)
      WRITE(6,25)
      WRITE(6,21)
      WRITE(6,26)
      WRITE(6,27)
      WRITE(6,28)
      WRITE(6,21)
      WRITE(6,6)
      WRITE(6,7)TIM(1),TIM(2),TIM(3),TIM(4),TIM(5),TIM(6),TIM(7)
      WRITE(6,21)
      DO 220 I=1,21
      IN=-11+I
      WRITE(6,8)IN,DI(I,1),DI(I,2),DI(I,3),DI(I,4),DI(I,5),DI(I,6)
     X,DI(I,7)
220   WRITE(6,9)
      WRITE(6,21)
      WRITE(6,35)
      WRITE(6,21)
      DO 221 I=1,21
      IN=-11+I
      WRITE(6,8)IN,DE(I,1),DE(I,2),DE(I,3),DE(I,4),DE(I,5),DE(I,6)
     X,DE(I,7)
221   WRITE(6,9)
      WRITE(6,21)
      WRITE(6,[illegible])
      WRITE(6,[illegible])
      WRITE(6,[illegible])VAR(1),VAR(2),VAR(3),VAR(4),VAR(5),VAR(6),VAR([illegible])
      CALL NTRAN(16,10)
      CALL NTRAN(16,1,441 ,AMS,L)
      CALL NTRAN(16,22)
      WRITE(6,3[illegible])
      WRITE(6,[illegible])L
      CALL NTRAN(17,10)
      CALL NTRAN(17,1,441 ,AIY,L)
      CALL NTRAN(17,22)
      WRITE(6,3[illegible])
      WRITE(6,[illegible])L
      CALL NTRAN(1[illegible],10)
      CALL NTRAN(1[illegible],1,441 ,AIZ,L)
      CALL NTRAN(1[illegible],22)
      WRITE(6,[illegible])
      WRITE(6,[illegible])L
      CALL NTRAN(19,10)
      CALL NTRAN(19,1,1,VA,L)
      CALL NTRAN(19,22)
      WRITE(6,79)
      WRITE(6,8[illegible])I
[illegible]9   CONTINUE
      END
```

```
C*************************************************************************
C
C
C           SECOND ORDER PHASE LOCK LOOP---GATED
C           SNR=9.0
C           INTEGRATER GAIN=AK
C           SECOND ORDER POLYNOMIAL APPROXIMATION
C           DELX=DELZ=PI/50        DELT=0.001
C           ONE FIFTH STORAGE
C
C
C*************************************************************************
      DIMENSION ANS(21,21),NANS(21,21),AIX(21,21),AIZ(21,21),POLX(2),
     XPOLZ(2),PDXZ(2),PDZX(2),VAR(50,35)
      REAL NANS
1     FORMAT(1H ,11X,5HX=-PI,4X,7HX=-.8PI,3X,7HX=-.6PI,3X,7HX=-.4PI,3X,
     X7HX=-.2PI,3X,3HX=0,7X,6HX=.2PI,4X,6HX=.4PI,4X,6HX=.6PI,4X,
     X6HX=.8PI,4X,4HX=PI)
2     FORMAT(1H ,I2,5X,11(F10.5))
3     FORMAT()
4     FORMAT(~~1H ,5HTIME=)~~
5     FORMAT(1H+,5X,F6.4)
6     FORMAT(1H ,4X,1HX,6X,2HT=,6(8X,2HT=))
7     FORMAT(1H+,3X,7(5X,F5.2))
8     FORMAT(1H ,I2,6X,7(F10.5))
9     FORMAT(1H+,3X,5HPI/10)
10    FORMAT(~~1H ,17HVARIANCE E~~(X**2)=)
11    FORMAT(1H ,8X,7(2X,F8.5))
15    FORMAT(1H ,2X,4HTIME,5(7X,5HCYCLE))
```

```
16     FORMAT(1H ,1X,F6.4,5(5X,F7.4))
17     FORMAT(1H ,4X,5(10X,I2))
20     FORMAT(1H ,75H***********************************************************
      X******************)
21     FORMAT(1H0,1H )
22     FORMAT(1H ,34HGATED SECOND ORDER PHASE LOCK LOOP)
23     FORMAT(1H ,9HSN/=9.0   )
24     FORMAT(1H ,18HINTEGRATER GAIN=AK)
25     FORMAT(1H ,37HSECOND ORDER POLYNOMIAL APPROXIMATION)
26     FORMAT(1H ,17HONE FIFTH STORAGE)
27     FORMAT(1H ,21HDISTRIBUTION FUNCTION)
28     FORMAT(1H ,16HDENSITY FUNCTION)
29     FORMAT(1H+,2X,4HPI/5)
30     FORMAT(1H1,1H )
31     FORMAT(1H ,32HDELX=DELZ=PI/50       DELT=0.001 )
32     FORMAT(1H ,10X,1HX,10X,4HF(X),15X,1HZ,10X,4HF(Z))
33     FORMAT(1H ,6X,I3,10X,F10.5,7X,I3,10X,F10.5)
34     FORMAT(1H+,9X,5HPI/10,25X,5HPI/10)
35     FORMAT(1H ,I10)
36     FORMAT(1H ,28HANS WAS NOT COPIED CORRECTLY)
37     FORMAT(1H ,28HAIX WAS NOT COPIED CORRECTLY)
38     FORMAT(1H ,28HAIZ WAS NOT COPIED CORRECTLY)
39     FORMAT(1H ,23HNUMBER OF WORDS STORED=)
42     FORMAT(1H+,25X,I10)
       DELX=0.062831853
      DELZ=0.062831853
      DELT=0.001
      PS=5.0
      SNR=9.0
      NI=1
      NF=10
      PSX=PS*DELX
      PSZ=PS*DELZ
      II=(6.2831853+PSX)/PSX+0.5
      JJ=(6.2831853+PSZ)/PSZ+0.5
      III=II-1
      JJJ=JJ-1
      IM2=II-2
      A=DELT/(2*DELX)
      B=DELT/(2*SNR*DELX**2)
      C=DELT/(2*DELZ)
      D=1.0-2*B
      E=1.0-B
      F=1/(2*DELX)
      G=1/(2*DELZ)
      CALL MTRAN(26,2,441 ,ABS,L)
      CALL MTRAN(26,22)
      IF(L.NE.441 )GO TO 90
      GO TO 91
90    WRITE(6,36)
      WRITE(6,35)L
      GO TO 99
91    CALL MTRAN(27,2,441 ,ATX,L)
      CALL MTRAN(27,22)
      IF(L.NE.441 )GO TO 92
      GO TO 93
92    WRITE(6,37)
      WRITE(6,35)L
      GO TO 99
93    CALL MTRAN(28,2,441 ,ATZ,L)
      CALL MTRAN(28,22)
      IF(L.NE.441 )GO TO 94
      GO TO 95
94    WRITE(6,38)
      WRITE(6,35)L
      GO TO 99
95    CALL MTRAN(29,2,1,VA,L)
      CALL MTRAN(29,22)
      IF(L.NE.1) GO TO 96
      GO TO 97
96    WRITE(6,38)
      WRITE(6,35)L
      GO TO 99
97    CONTINUE
      VA1=VA
      VAR(1,1)=VA1
      DO 80 N=NI,NE
      VAR(N,1)=VAR(N-1,21)
      DO 81 LL=2,6
      DO 75 L=1,25
      DO 40 J=2,JJJ
      Z=-3.1415927+(J-1)*PSZ
      DO 41 I=2,III
      X=-3.1415927+(I-1)*PSX
      FOX=ANS(I-1,J)
      DFOX=ANS(I,J)-ANS(I-1,J)
      D2FOX=ANS(I+1,J)-ANS(I,J)-DFOX
      FOZ=ANS(I,J-1)
      DFOZ=ANS(I,J)-ANS(I,J-1)
      D2FOZ=ANS(I,J+1)-ANS(I,J)-DFOZ
      DXZ=AIZ(I-1,J)
      DDXZ=AIZ(I,J)-AIZ(I-1,J)
      D2DXZ=AIZ(I+1,J)-AIZ(I,J)-DDXZ
      DZX=AIX(I,J-1)
      DDZX=AIX(I,J)-AIX(I,J-1)
      D2DZX=AIX(I,J+1)-AIX(I,J)-DDZX
      POLX(1)=FOX+0.8*DFOX-0.08*D2FOX
      POLX(2)=FOX+1.2*DFOX+0.12*D2FOX
      POLZ(1)=FOZ+0.8*DFOZ-0.08*D2FOZ
      POLZ(2)=FOZ+1.2*DFOZ+0.12*D2FOZ
      PDXZ(1)=DXZ+0.8*DDXZ-0.08*D2DXZ
      PDXZ(2)=DXZ+1.2*DDXZ+0.12*D2DXZ
      PDZX(1)=DZX+0.8*DDZX-0.08*D2DZX
      PDZX(2)=DZX+1.2*DDZX+0.12*D2DZX
      NANS(I,J)=POLX(2)*(Z*A-X*A+SIN(X)*A+B)+POLX(1)*(-Z*A+X*A-SIN(X)*A
     X+B)+POLZ(2)*(Z*C-X*C)+POLZ(1)*(-Z*C+X*C)+ANS(I,J)*D+PDZX(2)*C
     X-PDZX(1)*C-PDXZ(2)*A+PDXZ(1)*A
41    CONTINUE
      I=II
      X=-3.1415927+(I-1)*PSX
      FOZ=ANS(I,J-1)
      DFOZ=ANS(I,J)-ANS(I,J-1)
      D2FOZ=ANS(I,J+1)-ANS(I,J)-DFOZ
      DZX=AIX(I,J-1)
      DDZX=AIX(I,J)-AIX(I,J-1)
      D2DZX=AIX(I,J+1)-AIX(I,J)-DDZX
      POLZ(1)=FOZ+0.8*DFOZ-0.08*D2FOZ
      POLZ(2)=FOZ+1.2*DFOZ+0.12*D2FOZ
      PDZX(1)=DZX+0.8*DDZX-0.08*D2DZX
      PDZX(2)=DZX+1.2*DDZX+0.12*D2DZX
      NANS(I,J)=ANS(I,J)+POLZ(2)*(Z*C-X*C)+POLZ(1)*(-Z*C+X*C)+PDZX(2)*C
     X-PDZX(1)*C
 40   CONTINUE
      J=JJ
      Z=-3.1415927+(J-1)*PSZ
      DO 65 I=2,III
      X=-3.1415927+(I-1)*PSX
      FOX=ANS(I-1,J)
      DFOX=ANS(I,J)-ANS(I-1,J)
      D2FOX=ANS(I+1,J)-ANS(I,J)-DFOX
      DXZ=AIZ(I-1,J)
      DDXZ=AIZ(I,J)-AIZ(I-1,J)
      D2DXZ=AIZ(I+1,J)-AIZ(I,J)-DDXZ
      POLX(1)=FOX+0.8*DFOX-0.08*D2FOX
      POLX(2)=FOX+1.2*DFOX+0.12*D2FOX
      PDXZ(1)=DXZ+0.8*DDXZ-0.08*D2DXZ
      PDXZ(2)=DXZ+1.2*DDXZ+0.12*D2DXZ
 65   NANS(I,J)=POLX(2)*(Z*A-X*A+SIN(X)*A+B)+POLX(1)*(-Z*A+X*A-SIN(X)*A
     X+B)+ANS(I,J)*D-PDXZ(2)*A+PDXZ(1)*A
```

```
      NANS(II,JJ)=1.0
      DO 48 J=2,JJ
      DO 48 I=2,II
48    ANS(I,J)=NANS(I,J)
      DO 61 J=1,JJ
      AIX(1,J)=0.0
      AIX(2,J)=ANS(2,J)*PSX*0.5
61    AIX(3,J)=(ANS(3,J)+4.0*ANS(2,J))*PSX*0.333333
      DO 49 I=4,III,2
      DO 49 J=1,JJ
      AIX(I,J)=AIX(I-2,J)+(ANS(I,J)+4.0*ANS(I-1,J)+ANS(I-2,J))*PSX*0.333
     X3333
49    AIX(I+1,J)=AIX(I-1,J)+(ANS(I+1,J)+4.0*ANS(I,J)+ANS(I-1,J))*PSX*0.3
     X33333
      DO 62 I=1,II
      AIZ(I,1)=0.0
      AIZ(I,2)=ANS(I,2)*PSZ*0.5
62    AIZ(I,3)=(ANS(I,3)+4.0*ANS(I,2))*PSZ*0.333333
      DO 50 J=4,JJJ,2
      DO 50 I=1,II
      AIZ(I,J)=AIZ(I,J-2)+(ANS(I,J)+4.0*ANS(I,J-1)+ANS(I,J-2))*PSZ*0.333
     X3333
50    AIZ(I,J+1)=AIZ(I,J-1)+(ANS(I,J+1)+4.0*ANS(I,J)+ANS(I,J-1))*PSZ*0.3
     X33333
75    CONTINUE
      VA=0.0
      DO 200 I=1,IM2,2
200   VA=VA+ANS(I,JJ)*(3.1415927-(I-1)*PSX)+4.0*ANS(I+1,JJ)*
     X(3.1415927-I*PSX)+ANS(I+2,JJ)*(3.1415927-(I+1)*PSX)
```

```
      VAR(N,LL)=3.1415927**2+2.0*VA*PSX/3.0
81    CONTINUE
      DO 110 LL=7,21
      DO 111 L=1,25
      DO 340 J=2,JJJ
      Z=-3.1415927+(J-1)*PSZ
      DO 341 I=2,III
      X=-3.1415927+(I-1)*PSX
      FOX=ANS(I-1,J)
      DFOX=ANS(I,J)-ANS(I-1,J)
      D2FOX=ANS(I+1,J)-ANS(I,J)-DFOX
      FOZ=ANS(I,J-1)
      DFOZ=ANS(I,J)-ANS(I,J-1)
      D2FOZ=ANS(I,J+1)-ANS(I,J)-DFOZ
      DXZ=AIZ(I-1,J)
      DDXZ=AIZ(I,J)-AIZ(I-1,J)
      D2DXZ=AIZ(I+1,J)-AIZ(I,J)-DDXZ
      DZX=AIX(I,J-1)
      DDZX=AIX(I,J)-AIX(I,J-1)
      D2DZX=AIX(I,J+1)-AIX(I,J)-DDZX
      POLX(1)=FOX+0.8*DFOX-0.08*D2FOX
      POLX(2)=FOX+1.2*DFOX+0.12*D2FOX
      POLZ(1)=FOZ+0.8*DFOZ-0.08*D2FOZ
      POLZ(2)=FOZ+1.2*DFOZ+0.12*D2FOZ
      PDXZ(1)=DXZ+0.8*DDXZ-0.08*D2DXZ
      PDXZ(2)=DXZ+1.2*DDXZ+0.12*D2DXZ
      PDZX(1)=DZX+0.8*DDZX-0.08*D2DZX
      PDZX(2)=DZX+1.2*DDZX+0.12*D2DZX
      NANS(I,J)=POLX(2)*(Z*A-X*A+B)+POLX(1)*(-Z*A+X*A
     X+B)+POLZ(2)*(Z*C-X*C)+POLZ(1)*(-Z*C+X*C)+ANS(I,J)+D+PDZX(2)*C
     X-PDZX(1)*C-PDXZ(2)*A+PDXZ(1)*A
341   CONTINUE
      I=II
      X=-3.1415927+(I-1)*PSX
      FOZ=ANS(I,J-1)
      DFOZ=ANS(I,J)-ANS(I,J-1)
      D2FOZ=ANS(I,J+1)-ANS(I,J)-DFOZ
      DZX=AIX(I,J-1)
      DDZX=AIX(I,J)-AIX(I,J-1)
      D2DZX=AIX(I,J+1)-AIX(I,J)-DDZX
      POLZ(1)=FOZ+0.8*DFOZ-0.08*D2FOZ
      POLZ(2)=FOZ+1.2*DFOZ+0.12*D2FOZ
      PDZX(1)=DZX+0.8*DDZX-0.08*D2DZX
      PDZX(2)=DZX+1.2*DDZX+0.12*D2DZX
      NANS(I,J)=ANS(I,J)+POLZ(2)*(Z*C-X*C)+POLZ(1)*(-Z*C+X*C)+PDZX(2)*C
     X-PDZX(1)*C
340   CONTINUE
      J=JJ
      Z=-3.1415927+(J-1)*PSZ
      DO 365 I=2,III
      X=-3.1415927+(I-1)*PSX
      FOX=ANS(I-1,J)
      DFOX=ANS(I,J)-ANS(I-1,J)
      D2FOX=ANS(I+1,J)-ANS(I,J)-DFOX
      DXZ=AIZ(I-1,J)
      DDXZ=AIZ(I,J)-AIZ(I-1,J)
      D2DXZ=AIZ(I+1,J)-AIZ(I,J)-DDXZ
      POLX(1)=FOX+0.8*DFOX-0.08*D2FOX
      POLX(2)=FOX+1.2*DFOX+0.12*D2FOX
      PDXZ(1)=DXZ+0.8*DDXZ-0.08*D2DXZ
      PDXZ(2)=DXZ+1.2*DDXZ+0.12*D2DXZ
305   NANS(I,J)=POLX(2)*(Z*A-X*A+B)+POLX(1)*(-Z*A+X.A
     X+B)+ANS(I,J)*B-PDXZ(2)*A+PDXZ(1)*A
      NANS(I1,JJ)=1.0
      DO 348 J=2,JJ
      DO 348 I=2,II
348   ANS(I,J)=NANS(I,J)
      DO 361 J=1,JJ
      AIX(1,J)=0.0
      AIX(2,J)=ANS(2,J)*PSX*0.5
361   AIX(3,J)=(ANS(3,J)+4.0*ANS(2,J))*PSX*0.333333
      DO 349 I=4,III,2
      DO 349 J=1,JJ
      AIX(I,J)=AIX(I-2,J)+(ANS(I,J)+4.0*ANS(I-1,J)+ANS(I-2,J))*PSX*0.333
     X3333
349   AIX(I+1,J)=AIX(I-1,J)+(ANS(I+1,J)+4.0*ANS(I,J)+ANS(I-1,J))*PSX*0.3
     X33333
      DO 362 I=1,II
      AIZ(I,1)=0.0
      AIZ(I,2)=ANS(I,2)*PSZ*0.5
362   AIZ(I,3)=(ANS(I,3)+4.0*ANS(I,2))*PSZ*0.3333333
      DO 350 J=4,JJJ,2
      DO 350 I=1,II
      AIZ(I,J)=AIZ(I,J-2)+(ANS(I,J)+4.0*ANS(I,J-1)+ANS(I,J-2))*PSZ*0.333
     X3333
```

```
350   AIZ(I,J+1)=AIZ(I,J-1)+(ANS(I,J+1)+4.0*ANS(I,J)+ANS(I,J-1))*PSZ*0.3
     X333333
111    CONTINUE
       VA=0.0
       DO 400 I=1,IM2,2
400    VA=VA+ANS(I,JJ)*(3.1415927-(I-1)*PSX)+4.0*ANS(I+1,JJ)*
     X(3.1415927-I*PSX)+ANS(I+2,JJ)*(3.1415927-(I+1)*PSY)
       VAR(N,LL)=3.1415927**2+2.0*VA*PSX/3.0
       VA=VAR(N,L)
110    CONTINUE
80     CONTINUE
       VAR(N,1)=VA1
       WRITE(6,20)
       WRITE(6,21)
       WRITE(6,22)
       WRITE(6,23)
       WRITE(6,24)
       WRITE(6,25)
       WRITE(6,31)
       WRITE(6,2A)
       WRITE(6,21)
       WRITE(6,20)
       WRITE(6,21)
       DO 115 N=1,NF,5
       N1=N
       N2=N+1
       N3=N+2
       N4=N+3
       N5=N+4
       WRITE(6,21)
       WRITE(6,21)
```

```
      WRITE(6,15)
      WRITE(6,17)N1,N2,N3,N4,N5
      DO 116 J=1,21
      T=(J-1)*0.025
116   WRITE(6,16)T,VAR(N1,J),VAR(N2,J),VAR(N3,J),VAR(N4,J),VAR(N5,J)
115   CONTINUE
      CALL MTRAN(26,10)
      CALL MTRAN(26,1,441 ,AMS,L)
      CALL MTRAN(26,22)
      WRITE(6,39)
      WRITE(6,42)L
      CALL MTRAN(27,10)
      CALL MTRAN(27,1,441 ,ATX,L)
      CALL MTRAN(27,22)
      WRITE(6,39)
      WRITE(6,42)L
      CALL MTRAN(28,10)
      CALL MTRAN(28,1,441 ,ATZ,L)
      CALL MTRAN(28,22)
      WRITE(6,39)
      WRITE(6,42)L
      CALL MTRAN(29,10)
      CALL MTRAN(29,1,1,VA,L)
      CALL MTRAN(29,22)
      WRITE(6,39)
      WRITE(6,42)L
99    CONTINUE
      END
```

BIBLIOGRAPHY

BIBLIOGRAPHY

1. A. Ralston and H. S. Wilf, Mathematical Methods for Digital Computers, John Wiley and Sons, Inc., New York, Tenth Printing, 1967.

2. D. Middleton, An Introduction to Statistical Communication Theory, McGraw-Hill, New York, 1960.

3. A. Papoulis, The Fourier Integral and Its Applications, McGraw-Hill, New York, 1962.

4. E. B. Dynkin, Markov Processes, Springer-Verlag, Berlin, 1965.

5. A. R. Mitchell, Computational Methods in Partial Differential Equations, John Wiley and Sons, Inc., New York, 1969.

6. P. A. P. Morgan, An Introduction to Probability Theory, Clarendon Press, Oxford, 1968.

7. F. H. Miller, Partial Differential Equations, John Wiley and Sons, Inc., Seventh Printing, New York, 1953.

8. A. J. Viterbi, Principles of Coherent Communications, McGraw-Hill, New York, 1966.

9. F. M. Gardner, Phaselock Techniques, John Wiley and Sons, Inc., New York, 1966.

10. A. T. Bharucha-Reid, Elements of the Theory of Markov Processes and Their Application, McGraw-Hill, New York, 1960.

11. R. F. Pawula, "Generalizations and Extensions of the Fokker-Planck-Kolmogorov Equations," IEEE Transactions on Information Theory, Vol. IT-13, No. 1, January 1967, p. 33-41.

12. K. E. Dominiak and R. L. Pickholtz, "Transient Behavior of a Phase-Lock Loop in the Presence of Noise," IEEE Transactions on Communication Technology, Vol. 18, No. 4, August 1970, p. 452-456.

13. R. M. Howe and S. K. Hsu, "Preliminary Investigation of a Hybrid Method for Solving Partial Differential Equations," Applied Dynamics, Inc., Ann Arbor, Michigan, October, 1967.

BIBLIOGRAPHY (Continued)

14. D. M. Young and T. G. Frank, "A Survey of Computer Methods for Solving Elliptic and Parabolic Partial Differential Equations," ICC Bulletin, Vol. 2, No. 1, January 1963, p. 3-61.

15. J. C. Whitney, "Finite Difference Methods for the Fokker-Planck Equation," Report No. 45, Plasma Laboratory, School of Engineering and Applied Science, Columbia University, May 1969.

16. J. L. Doob, "The Brownian Movement and Stochastic Equations," Annals of Math, Vol. 43, No. 2, 1942, p. 351-369.

17. W. Feller, An Introduction to Probability Theory and Its Applications, Vol. 2, John Wiley and Sons, Inc., New York, 1966.

18. J. L. Doob, Stochastic Processes, John Wiley and Sons, Inc., New York, 1953.

19. R. L. Stratonovich, Topics in the Theory of Noise, Vol. 1, Second Printing, Gordon and Breach, New York, 1967.

20. B. V. Gnedenko, The Theory of Probability, Chelsea Publishing Co., New York, Fourth Edition, 1967.

21. K. Ito, "On Stochastic Differential Equations," Memoirs of the American Math. Society, No. 4, 1951.

22. J. vonNeumann and R. D. Richtmyer, "On the Numerical Solution of Partial Differential Equations of the Parabolic Type," Los Alamos Rept. LA657, December 1947.

23. D. L. Snyder, "The State-Variable Approach to Continuous Estimation with Applications to Analog Communication Theory," M.I.T. Research Monograph Series, Vol. 51, M.I.T. Press, Cambridge, Mass., 1969.

24. P. J. Davis, Interpolation and Approximation, Blaisdell Publishing Co., New York, 1963.

25. E. Isaacson and H. B. Keller, Analysis of Numerical Methods, John Wiley and Sons, Inc., New York, 1966.

26. F. J. Charles and W. C. Lindsey, "Some Analytical and Experimental Phase-Locked Loop Results for Low Signal-to-Noise Ratios," IEEE Proceedings, Vol. 54, No. 9, September 1966, p. 1152-1166.

27. W. W. Mayfield, "Nonlinear System Analysis by Fokker-Planck Techniques with Applications to Frequency Modulation Tracking," Ph.D. Thesis, University of Southern California, August 1970.

BIBLIOGRAPHY (Continued)

28. J. R. La Frieda, "Transient Solution of the Fokker-Planck Equation for a Class of First Order Phase-Lock Loops," 1971 Proceedings of the Fourth Hawaii International Conference on System Sciences, Western Periodicals Co., 1971, p. 471-473.

29. G. E. Uhlenbeck and L. S. Ornstein, "On the Theory of the Brownian Motion," Physical Review, Vol. 36, September 1930.

30. D. L. Finn, "Bibliography on Techniques for Solving Partial Differential Equations by Hybrid Computation and Other Methods," Technical Note No. 16, Georgia Tech Project A-588, Marshall Space Flight Center Contract No. NAS8-2473, April 1968.

31. W. M. O'Dowd and J. L. Hammond, Jr., "A Preliminary Study of Methods for Solving Partial Differential Equations with a Hybrid Computer," Technical Note No. 21, Georgia Tech Research Project A-588, Marshall Space Flight Center Contract No. NAS8-2473, September 1970.

32. A. Papoulis, Probability, Random Variables, and Stochastic Processes, McGraw-Hill, New York, 1965.

33. H. L. Van Trees, Detection, Estimation, and Modulation Theory, Part II, John Wiley and Sons, Inc., New York, 1971.

34. R. Vitchnevetsky, "State of the Art in Hybrid Methods for Partial Differential Equations," Electronic Associates, Inc., and Princeton University, Princeton, N. J., September 1970.

35. R. Vitchnevetsky, "Analog/Hybrid Computer Methods for Partial Differential Equations," Electronic Associates, Inc., and Princeton University, Princeton, N. J., October 1969.

36. R. M. Howe, "Solution of Partial Differential Equations," Lecture notes for the short course "Hybrid Computer Solution of Partial Differential Equations," Georgia Institute of Technology, October 1969.

37. J. K. Holmes, "On a Solution to the Second-Order Phase-Locked Loop," IEEE Transactions of Communication Technology, Vol. com-18, No. 2, April 1970, p. 119-126.

38. H. L. Van Trees, Detection, Estimation, and Modulation Theory, Part I, John Wiley and Sons, Inc., New York, 1968.

39. M. Loève, Probability Theory, D. Van Nostrand Co., Inc., Princteon, N. J., 1960.

BIBLIOGRAPHY (Concluded)

40. C. O. Alford, "Introduction to Partial Differential Equations and Survey of Solutions Methods," Lecture notes for the short course "Hybrid Computer Solution of Partial Differential Equations," Georgia Institute of Technology, October 1969.

41. W. C. Lindsey, Synchronization Systems in Communication and Control, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1972.

42. A. I. Rubin, "Hybrid Techniques for Generation of Arbitrary Functions," Simulation, Vol. 7, No. 6, December 1966.

43. A. N. Kolmogorov, "On Analytical Methods in Probability Theory," Annals of Math., Vol. 104, p. 415-458 (in German), 1931.

44. R. C. Buck, Advanced Calculus, McGraw-Hill, New York, 1965.

45. R. Tansworthe, "Cycle Slipping in Phase-Locked Loops," IEEE Transactions on Communication Technology, Vol. com-15, No. 3, June 1967, pp. 417-421.

46. F. Grandoni and U. Mengale, "Transient Phenomena in a Phase-Locked Loop with a Noisy Reference," Record of the IEEE 1972 National Telecommunications Conference, p. 32E-1.

47. H. I. Paul and W. E. Larimore, "Gated Phase-Locked Loop in the Presence of Noise," Record of the IEEE 1972 International Conference on Communications, p. 6-12.

VITA

William M. O'Dowd, Jr., was born July 15, 1943, in Augusta, Georgia. He is the son of Margaret M. and William M. O'Dowd, Sr. In March 1973, he married the former Connie L. Davis of Atlanta, Georgia.

Mr. O'Dowd attended parochial elementary and secondary schools in Augusta, Georgia. He attended the Georgia Institute of Technology as a cooperative student and received a B.S. degree in E.E. in 1966. He received a M.S. degree in E.E. from the Georgia Institute of Technology in 1967.