1983

Time-independent solutions to the Field-Noyes model of the diffusing Belousov-Zhabotinskii reaction

Paul K. Becker
The University of Montana

TIME-INDEPENDENT SOLUTIONS TO THE FIELD-NOYES MODEL OF THE DIFFUSING BELOUSOV-ZHABOTINSKII REACTION

by

Paul K. Becker

B.A., University of Chicago, 1968

M.A., University of Chicago, 1973

Presented in partial fulfillment of the requirements

for the degree of

Master of Science

UNIVERSITY OF MONTANA

1983

Approved by:

Richard J. Heyboer
Co-Chairman, Board of Examiners

Richard J Field
Co-Chairman, Board of Examiners

Dean, Graduate School

8-8-83
Date

ABSTRACT

Becker, Paul K., M.S., August 1983 Physics

Time-Independent Solutions to the Field-Noyes Model of the Diffusing Belousov-Zhabotinskii Reaction (118 pp.)

Directors: Richard J. Hayden RJH Richard J. Field R.J.F.

The Belousov-Zhabotinskii (BZ) reaction is an unusual chemical phenomenon which, when far from chemical equilibrium, can display oscillations in the concentrations of some intermediate species. A simplified model for this reaction was constructed by Field and Noyes. The behavior of this reaction model in space and time is governed by kinetic equations coupled with diffusion terms. The reaction-diffusion model leads to three partial differential equations in three variables. The purpose of this numerical study is to find stable solutions to these equations by manipulating initial conditions and the parameters f and D_z. A GEARB numerical integrator is used to trace the evolution of waveforms and to test for stability.

The reaction-diffusion equations have a unique zeroth-order solution. Perturbation analysis is used to show that small first and second-order solutions exist for certain critical lengths. Small perturbations at these critical lengths develop on the integrator into waveforms resembling the fundamental first-order solutions. These waveforms are quasi-stable while at small amplitudes.

Prior to seeking large-amplitude stable waveforms, x-y phase space is investigated for a range of z values. The $x''=0$ and $y''=0$ lines in this phase space are made to intersect in three points by adjusting f and D_z. Two of the intersection points are stable nodes. Using these nodes to set initial conditions, large-amplitude time-independent solutions are found for a range of f from 0.1 to 2.5 and for D_z from 0.1 to 10×10^{-5} cm sec^{-1}, the latter case only occurring when f=0.6. A set of necessary conditions is proposed for the existence of large-scale solutions for a given set of parameters. The evolution of a small-amplitude, quasi-static solution into a stable large solution is shown. The minimum factor of ten between diffusion rates necessary for stability is physically reasonable for macromolecules, but not for the BZ reaction. The possibility of further reducing D_z is discussed.

ACKNOWLEDGMENTS

I would like to thank Dr. Richard J. Field, Physical Chemistry, and Dr. Richard J. Hayden, Physics and Astronomy, for their special forms of advice, support and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

A. PROBLEM

The metal-ion-catalyzed oxidation and bromination of certain organic compounds by bromate ions is referred to as the Belousov-Zhabotinskii (BZ) reaction[1]. During the course of the BZ reaction, temporal oscillations are observed in the ratio of the concentration of the oxidized to the reduced form of the catalyst and in the bromide ion concentration. A typical experimental result is shown in Fig. 1.1. The concentrations of other species oscillate but are not readily measured. A group of eighteen chemical reactions describing this complex oscillating system was proposed by Field, Körös and Noyes[2] in 1972. In 1974 Field and Noyes[3] suggested a reduced model for this system consisting of five chemical reactions. The kinetic equations for these reactions are three ordinary differential equations with time as the independent variable and three chemical concentrations as the dependent variables. When evaluated numerically, the three kinetic equations stemming from the Field-Noyes model (known as the Oregonator) produce oscillatory behavior qualitatively similar to that displayed experimentally by the BZ reaction. In addition, the Oregonator, when coupled with diffusion terms, can numerically reproduce traveling waves of chemical activity[4], again successfully modeling the BZ reaction[5,6]. The Oregonator is now widely accepted as a good working approximation of

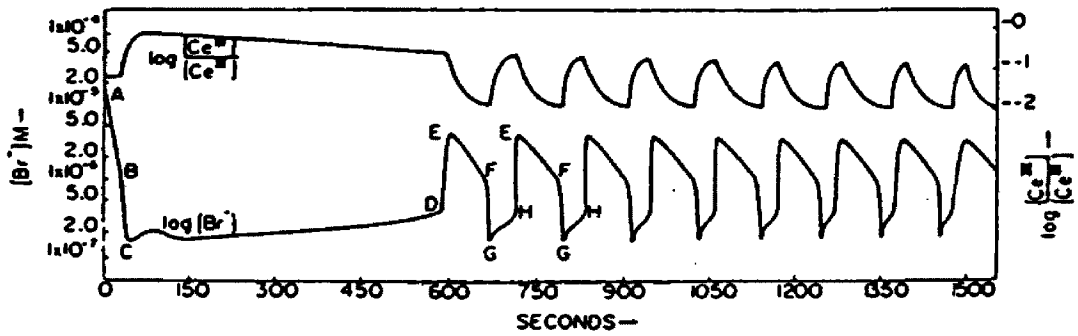


Figure 1.1 Potentiometric traces of log [Br^-] and log [Ce(IV)]/[Ce(III)] for a representative reaction exhibiting all six periods. Initial concentrations were $[CH_2(COOH)_2]_0 = 0.032\ M$, $[KBrO_3]_0 = 0.063\ M$, $[KBr]_0 = 1.5 \times 10^{-5}\ M$, $[Ce(NH_4)_2(NO_3)_5]_0 = 0.001\ M$, $[H_2SO_4]_0 = 0.8\ M$.

the BZ reaction.

As early as 1952 Turing[7] suggested that autocatalytic chemical kinetics coupled with diffusion could generate spatial patterns of chemical concentrations. He proposed that such spatial chemical arrangements, if they changed slowly, could be a mechanism for controlling morphogenesis, a process in which a symmetric group of cells grows into an asymmetric shape. Nicolis and Prigogine[8] have found such stable spatial solutions to a set of nonlinear partial differential equations arising from an autocatalytic chemical model known as the Brusselator. The Brusselator model, though presented in the form of chemical equations, does not correspond to an actual chemical system. A stable, spatial concentration gradient supported by the free energy change in a chemical reaction is a most unusual occurrence. The goal of this thesis is to find stable solutions to the reaction-diffusion Oregonator, a model which does have a real chemical basis.

B. EQUATIONS

The Oregonator model can be written as

$$A + Y \xrightarrow{k_1} X$$

$$X + Y \xrightarrow{k_2} P$$

$$B + X \xrightarrow{k_3} 2X + Z$$

$$2X \xrightarrow{k_4} Q$$

$$Z \xrightarrow{k_5} fY.$$

where $A \equiv B \equiv BrO^-$, $X \equiv HBrO_2$, $Y \equiv Br^-$ and $Z \equiv Ce(IV)$. The letters P and Q represent generalized products of the reaction whose concentrations do not enter into the kinetic equations. Each of the symbols k_1 to k_5 is a rate constant, the proportionality constant between the rate of a particular reaction and the concentration of its reactants. The symbol f is a stoichiometric factor in the fifth equation, representing the number of Y's which appear for every Z which disappears in the reaction. The value of f can physically range from 0 to about 5. The chemical species X, Y and Z have diffusion coefficients D_x, D_y and D_z, respectively. Commonly X, Y and Z are changed to dimensionless variables x, y and z. For these variables the equations describing the dynamics of a system with one spatial dimension are

$$(\partial x/\partial t)|_\ell = D_x'(\partial^2 x/\partial \ell^2)|_t + s(y - xy + x - qx^2) \qquad \text{Eq. 1.2}$$

$$(\partial y/\partial t)|_\ell = D_y'(\partial^2 y/\partial \ell^2)|_t + (-y - xy + fz)/s \qquad \text{Eq. 1.3}$$

$$(\partial z/\partial t)|_\ell = D_z'(\partial^2 z/\partial \ell^2)|_t + w(x - z) \qquad \text{Eq. 1.4}$$

where

$$x = \frac{k_1 X}{k_1 A} \qquad y = \frac{k_2 Y}{k_3 B} \qquad z = \frac{k_2 k_5 Z}{k_1 k_3 AB} \qquad t = \sqrt{k_1 k_3 AB}\, T$$

$$q = \frac{2k_1 k_4 A}{k_2 k_3 B} \qquad w = \frac{k_5}{\sqrt{k_1 k_3 AB}} \qquad s = \sqrt{\frac{k_3 B}{k_1 A}} \qquad D_i' = \frac{100 D_i}{\sqrt{k_1 k_3 AB}} .$$

For convenience the units of D_i have been changed from $cm^2 s^{-1}$ to $mm^2 s^{-1}$. Throughout this paper the rate constants currently recommended by Tyson[9] are used. These are referred to as the "Low" rate constants. See Appendix 1 for the values of rate constants and parameters used.

C. INTEGRATOR

Although time-independent solutions are sought for the Oregonator reaction-diffusion equations, a computerized integrator is needed which can solve the equations in both time and space in order to check these solutions for stability. The procedure used by Field and Reusser[4] to investigate traveling waves is followed to construct such an integrator. The method of lines is used to change from a set of partial differential equations to a much larger set of ordinary differential equations with time as the independent variable. The length of space to be investigated is broken into a large number of discrete points. Unless otherwise mentioned, 250 spatial points are used. The second spatial partial derivative for x is estimated by

$$\left.\frac{\partial^2 x(n)}{\partial \ell^2}\right|_t = \left.\frac{x(n+1) - x(n) - [x(n) - x(n-1)]}{(\Delta \ell)^2}\right|_t .$$

A parallel procedure is used for the y and z variables. With this

approximation, and 250 space points, there are 750 variables and 750 nonlinear, ordinary differential equations with time as the independent variable.

Attempts to numerically integrate these equations by a Runge-Kutta method have proved quite awkward because the system has a numerical difficulty called "stiffness"[10]. In stiff equations a small numerical error in one variable leads extremely quickly to a large-scale deviation from the solution being sought. Stiffness is a common difficulty in chemical kinetics problems where time scales of various reactions may differ by many orders of magnitude. In particular the term (y-xy) in Eq. 1.2 can cause the numerical value of $\partial x/\partial t$ to oscillate wildly at the point where x approaches 1 and y becomes very large, even if the step size for the integrator is made very small[3]. This problem is eliminated by using GEARB, a variation[11] of an integrator originally devised by Gear[12]. Consider the 750 variables as a vector over time. GEARB steps from time t to time $t+\Delta t$ by finding an incremented vector at $t+\Delta t$ which has the property that when its total time derivative is computed and used to project back to time t, the original value of the vector at time t is hit. In practice the point at time $t+\Delta t$ is found by solving the set of 750 linear approximations to the 750 non-linear equations. The process involves inverting the Jacobian of the nonlinear ordinary differential equations. GEARB takes advantage of the banded nature of the Jacobian to integrate the equations with relative efficiency.

GEARB is initially supplied with values for each variable at every space point, a desired step size and error parameter, and subroutines which contain the differential equations and the Jacobian matrix. The Gear method works well for the Oregonator. All computations were done on the University of Montana DEC-20 computer. All numbers, with the exception of the Jacobian matrix, were kept in double precision mode. The DRIVEB version of GEARB was used. Appendix 3 contains the following main program and subroutines for the GEARB package: GOING, NSET, PARSET, PATT, INS, PDB, DIFFUN, PIC, PICTUR and RUNTIM.

D. BOUNDARY CONDITIONS: ONE SPATIAL DIMENSION

Keeping in mind the chemical nature of the equations, the two most likely boundary conditions for a one-dimensional closed space are: no diffusion at the edges, representing a container with impermeable walls, and constant chemical concentration at the edges, representing a permeable container in a large reservoir. This paper deals only with the former condition. The condition of no diffusion at the walls is equivalent to requiring that there be a zero slope for the chemical concentration curves at the edges.

CHAPTER II

QUASI-STABLE, SMALL-AMPLITUDE SOLUTIONS

A. INTRODUCTION

Any stationary solution of the Oregonator reaction-diffusion problem must satisfy the following three equations.

$$D_x'(d^2x/d\ell^2) = s(xy + qx^2 - y - x) \qquad \text{Eq. 2.1}$$

$$D_y'(d^2y/d\ell^2) = (y + xy - fz)/s \qquad \text{Eq. 2.2}$$

$$D_z'(d^2z/d\ell^2) = w(z - x) \qquad \text{Eq. 2.3}$$

Letting 0 and L denote the two endpoints of a chosen length, the boundary conditions are

$$(dx/d\ell)|_0 = (dy/d\ell)|_0 = (dz/d\ell)|_0 = 0$$
$$(dx/d\ell)|_L = (dy/d\ell)|_L = (dz/d\ell)|_L = 0 \qquad \text{Eq. 2.4}$$

If the second derivative terms are assumed to be zero, the above equations have a unique solution. That solution is relatively uninteresting here as it represents a container with chemical concentrations everywhere the same. Such a flat solution will be referred to as the zeroth-order solution. It may or may not be stable. The solutions sought in this work should have non-zero, stationary chemical gradients.

If one assumes the existence of a small-amplitude solution to Eqs. 2.1-2.4 that differs only slightly at each point from the zeroth-order solution, it is worthwhile to use a perturbation method to find first and second-order approximations. Since nothing is known about the stability in time of either of these approximations or of the solution they may approach, stability is checked by letting the GEARB integrator run with the first or second-order approximations as initial settings.

B. ZEROTH-ORDER SOLUTION

The zeroth-order solution to the time-independent kinetic equations is found by setting the second-order spatial derivative equal to zero in Eqs. 2.1, 2.2 and 2.3. Solving these three equations gives

$$x_o = [(1 - f - q) + \sqrt{(1 - f - q)^2 + 4q(f + 1)}\ ]/2q$$

$$y_o = fx_o/(1 + x_o)$$

$$z_o = x_o$$

C. FIRST-ORDER SOLUTION

Let x_s, y_s and z_s, each a function of ℓ, represent an assumed solution of the time-independent Oregonator reaction-diffusion equations which differs only slightly from the zeroth-order solution. Then define x*, y* and z* by the following equations and inequalities.

$$x_s = x_o + x^* \qquad y_s = y_o + y^* \qquad z_s = z_o + z^*$$

where

$x^*, y^*, z^* \ll 1$ and $x^*, y^*, z^* \ll x_s, y_s, z_s$.

By substitution into Eq. 2.2, the variable y^*, for example, then satisfies

$$D_y' [d^2(y_o + y^*)/d\ell^2] = [(y_o + y^*) + (x_o + x^*)(y_o + y^*) - f(z_o + z^*)]/s$$

or equivalently

$$D_y' [d^2y^*/d\ell^2] = [y^* + (x^*y_o + x_oy^* + x^*y^*) - fz^*]/s.$$

Differential equations for x^* and z^* are found similarly. Terms involving the product of two starred variables are small compared to other terms in these equations. If one lets x_1, y_1 and z_1 be new variables for a modification of the above equation (and the other two parallel equations) with any term involving the product of two starred variables thrown away, the following linear equations remain. Note that a number of terms involving zeroth-order variables have dropped out.

$$D_x' [d^2x_1/d\ell^2] = s(-y_1 + x_1y_o + x_oy_1 - x_1 + 2qx_ox_1)$$

$$D_y' [d^2y_1/d\ell^2] = (y_1 + x_1y_o + x_oy_1 - fz_1)/s \qquad \text{Eq. 2.5}$$

$$D_z' [d^2z_1/d\ell^2] = w(z_1 - x_1)$$

Define doubly starred variables so that

$$x^* = x_1 + x^{**} \qquad y^* = y_1 + y^{**} \qquad z^* = z_1 + z^{**}.$$

Solutions x^* and x_1 satisfy very similar differential equations. The only difference in the equations is the omission of terms involving products of starred variables. It is assumed that the contribution of

the omitted terms to the growth of each variable is small compared to that of the other terms. Thus x_1 should be close to x^*. Since $x^{**}=x^*-x_1$, it is expected that

$$x^{**} << x_1 \qquad y^{**} << y_1 \qquad z^{**} << z_1 \quad .$$

Equations 2.5 are linear second-order differential equations which can be solved. Assume solutions of the form

$$x_1(\ell) = x_a e^{ig\ell} \qquad y_1(\ell) = y_a e^{ig\ell} \qquad z_1(\ell) = z_a e^{ig\ell}$$

and substitute into the linearized equations, rearranging the equations slightly and putting the whole system into a matrix form.

$$\begin{pmatrix} -y_o+1-2qx_o & 1-x_o & 0 \\ -y_o & -1-x_o & f \\ 1 & 0 & -1 \end{pmatrix} \begin{pmatrix} x_a \\ y_a \\ z_a \end{pmatrix} = \begin{pmatrix} D'_x g^2/s & 0 & 0 \\ 0 & D'_y sg^2 & 0 \\ 0 & 0 & D'_z g^2/w \end{pmatrix} \begin{pmatrix} x_a \\ y_a \\ z_a \end{pmatrix} \qquad \text{Eq. 2.6}$$

This system is an eigenvalue problem in g^2. The program SMALL (see Appendix 3) drives an IMSL routine EIGZC which numerically solves this type of problem for complex matrix elements. SMALL provides the six values of g and, for each such value, the ratio of the coefficients, x_a, y_a and z_a which go with it. Thus the general solution for the linearized equations is of the form

$$x_1 = C_1(x_a e^{ig_a \ell}) + C_2(x_b e^{ig_b \ell}) + \ldots + C_6(x_f e^{ig_f \ell})$$
$$y_1 = C_1(y_a e^{ig_a \ell}) + C_2(y_b e^{ig_b \ell}) + \ldots + C_6(y_f e^{ig_f \ell}) \qquad \text{Eq. 2.7}$$
$$z_1 = C_1(z_a e^{ig_a \ell}) + C_2(z_b e^{ig_b \ell}) + \ldots + C_6(z_f e^{ig_f \ell})$$

with arbitrary constants C_1 to C_6.

This general solution must now be limited to fit the given boundary conditions: the first (spatial) derivative of each variable is equal to zero at ℓ=0 and at ℓ=L. To see the process of fitting the general solution to the boundary conditions more clearly, the general solution (Eq. 2.7) is rewritten in matrix form

$$\begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} = \begin{pmatrix} M_a^1 & M_b^1 & M_c^1 & M_d^1 & M_e^1 & M_f^1 \\ M_a^2 & M_b^2 & M_c^2 & M_d^2 & M_e^2 & M_f^2 \\ M_a^3 & M_b^3 & M_c^3 & M_d^3 & M_e^3 & M_f^3 \end{pmatrix} \begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \end{pmatrix}$$

or

$$\begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} = \underset{\sim}{M}_6^3 \begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \end{pmatrix}$$

where each M is a function of ℓ. Then the boundary conditions become

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = \left(\left.\frac{d}{d\ell}\right|_0 \quad \underset{\sim}{M}_6^3 \right) \begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \end{pmatrix}$$

and

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = \left(\left.\frac{d}{d\ell}\right|_L \quad \underset{\sim}{M}_6^3 \right) \begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \end{pmatrix} .$$

Combining in one large matrix

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} (d/d\ell)|_0 & (M_a^1 \ldots M_f^1) \\ (d/d\ell)|_0 & (M_a^2 \ldots M_f^2) \\ (d/d\ell)|_0 & (M_a^3 \ldots M_f^3) \\ (d/d\ell)|_L & (M_a^1 \ldots M_f^1) \\ (d/d\ell)|_L & (M_a^2 \ldots M_f^2) \\ (d/d\ell)|_L & (M_a^3 \ldots M_f^3) \end{pmatrix} \begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \end{pmatrix} . \qquad \text{Eq. 2.8}$$

In order to have a non-trivial solution to this matrix equation, the determinant of the six by six matrix must equal zero. SMALL finds the values of ℓ for which the determinant is zero. For L equal to one of these particular values, SMALL then gives the values of the C_i's, the relative amplitudes of the normal modes.

The results of this whole process are quite simple. The eigenvalue matrix equation (Eq. 2.6) is a third order equation in g^2, where g^2 is complex. Equation 2.8 can be solved if and only if at least one of the eigenvalues g^2 of Eq. 2.6 is real and positive. For each such real positive eigenvalue g_m^2 there is a critical L_m for which Eq. 2.8 is satisfied. For that L_m the coefficients of the two modes, $e^{ig_m\ell}$ and $e^{-ig_m\ell}$, are equal and the coefficients of the other modes are zero. Call that unique non-zero coefficient C/2. Then $(C/2)xe^{ig_m\ell}+(C/2)xe^{-ig_m\ell}=C[\cos(g_m\ell)]$. The only first-order solutions to

the time-independent Oregonator kinetic equations are a steady-state term plus a term proportional to a simple cosine function.

D. RESULTS FOR FIRST-ORDER SOLUTION

1. Note on a Numerical Problem

The kinetic equations for the basic Oregonator with no diffusion terms (i.e., in a well-mixed container) have a unique steady state. Using the method presented by Field and Noyes[3] and improved by Murray[13], one can compute that for the "Low" parameters (see Appendix 1) the steady state is unstable when $0.5024<f<2.3545$ and stable when f is outside this range (see Appendix 2). When the GEARB integrator is run with no diffusion terms and an initial condition only slightly perturbed from the steady state, and f is set at 1.82 (a value often used), the variables initially oscillate with a small amplitude. This amplitude slowly grows larger over a few cycles until large-amplitude oscillations appear. However, when the amplitude of the initial perturbation is made as small as 10^{-7}, only the eighth significant digit in the variables changes with time: the variables oscillate but do not grow larger. This tiny, bounded oscillation is a numerical problem with GEARB. Trouble occurs in the eighth digit even in double-precision calculations (16 significant digits) because very large, nearly equal, numbers are subtracted from each other, resulting in a loss of accuracy. When the equations for the basic Oregonator (with no diffusion terms) are perturbed by 10^{-5} or more, the solutions grow as expected.

Perturbations of amplitude roughly 10^{-7} for the Oregonator with diffusion are often used in this work. Such a small initial perturbation gives longer lasting waveforms. Over hundreds of integrations, the GEARB program got trapped at that tiny level only once, and that on an unimportant run. Nevertheless, to be sure that this problem has no deceptive effect on the estimate of the longevity of waveforms, the life of a waveform is denoted by the symbol Δt. The clock measuring Δt begins when the peak-to-peak amplitude of the x variable is roughly 0.001. At that time, the fourth significant digit in the value of x begins to change as one compares the left and right-hand sides of the wave. Values of x may get smaller again after the Δt clock is started, as even in the basic Oregonator (without diffusion) x may oscillate with small amplitude once more around the steady state before going large. The clock for Δt is stopped when the waveforms fly off to a large-scale shape. Numerically this condition is easy to recogize. In practice Δt is stopped when x reaches three times its steady state value.

2. Results for f=1.82

Using "Low" parameters (Appendix 1), the program SMALLK shows that small-scale cosine solutions exist only for $0.61<f<1.82$. That is, for these values of f, at least one of the eigenvalues g^2 is real and positive. With f=1.82 SMALL gives the following two eigenvalues for g^2.

$g^2 = 57.422558$	$g^2 = 28.622757$
$g = 7.5777673$	$g = 5.3500240$

First-order solutions are proportional to $\cos(g\ell)$, with L picked so $gL=n\pi$. For n=1

$L = 0.41458025$ mm. $\qquad$ $L = 0.58721093$ mm.

The associated eigenvectors for each value of g are

$x_a = C$ $\qquad$ $x_a = C$

$y_a = C(-0.22016735)$ $\qquad$ $y_a = C(-0.19541754)$

$z_a = C(0.14831832)$ $\qquad$ $z_a = C(0.25891472)$

where C is an arbitrary scaling constant. The solutions for g=5.3500240 are given below.

$$x_1 = C[\cos(g\ell)]$$

$$y_1 = C(-0.19541754)\cos(g\ell)$$

$$z_1 = C(0.25891472)\cos(g\ell)$$

Also, for f=1.82,

$$x_o = 3.4242442$$

$$y_o = 1.4086303$$

$$z_o = 3.4242442 \quad .$$

The first-order approximation for x_s is given by x_0+x_1. See Fig. 2.1 for graphs of these solutions. Note that in Figs. 2.1-2.3, L=0.58721093 even though the abscissa scale goes to 0.6.

The value of Δt for the basic Oregonator with no diffusion and initial conditions of Fig. 2.1 is roughly 25 time units, or 125 seconds in real time. When diffusion terms are added one finds the following Δt values for different size containers.

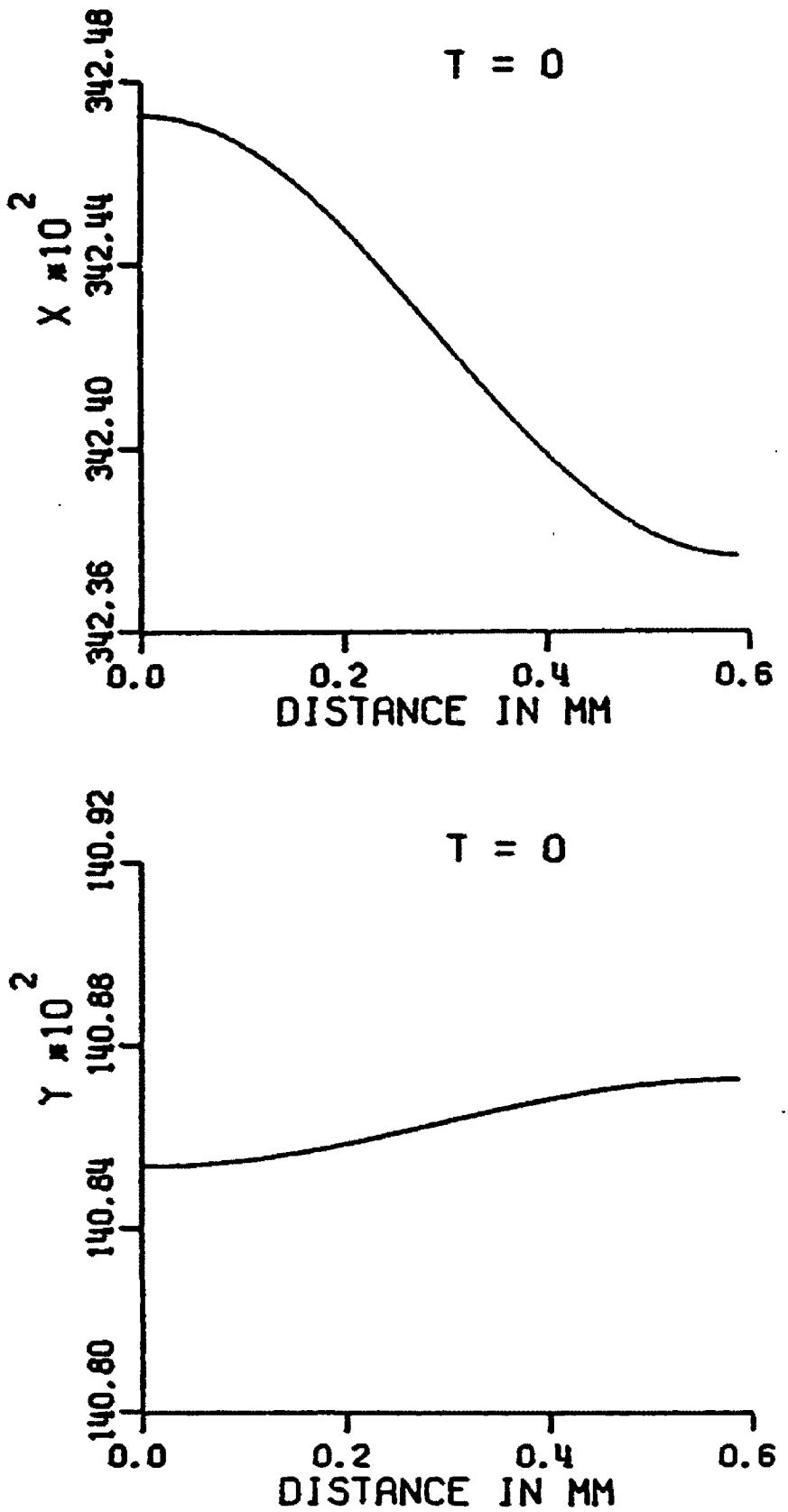


Figure 2.1: First-order solution. "Low" parameters
f=1.82 C=4.8 x 10^{-4} L=0.5872

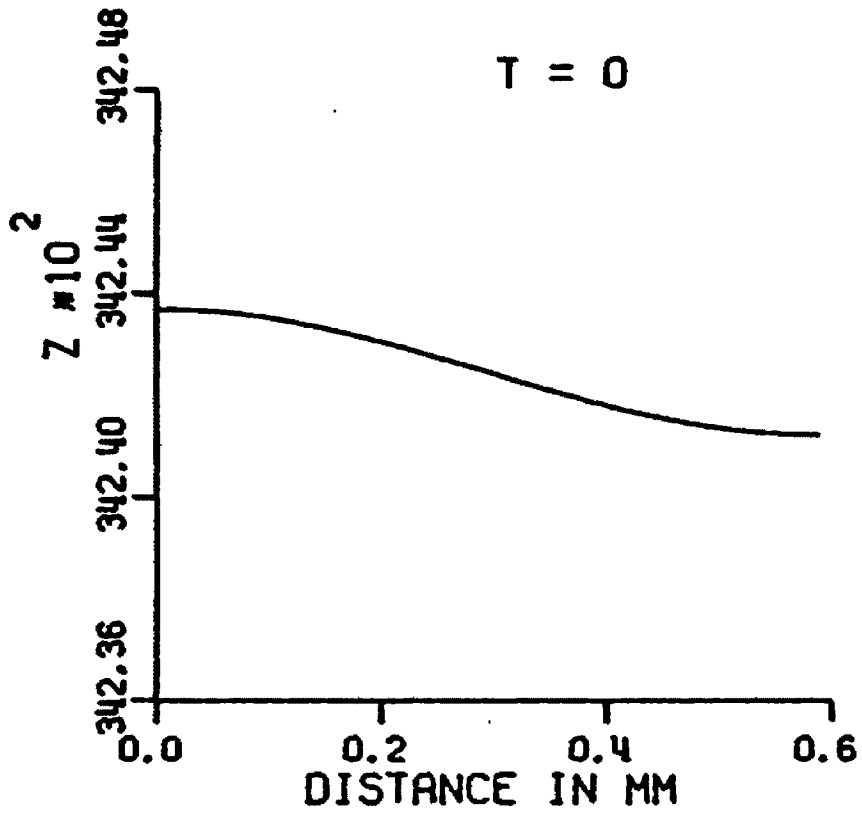


Figure 2.1 (continued)

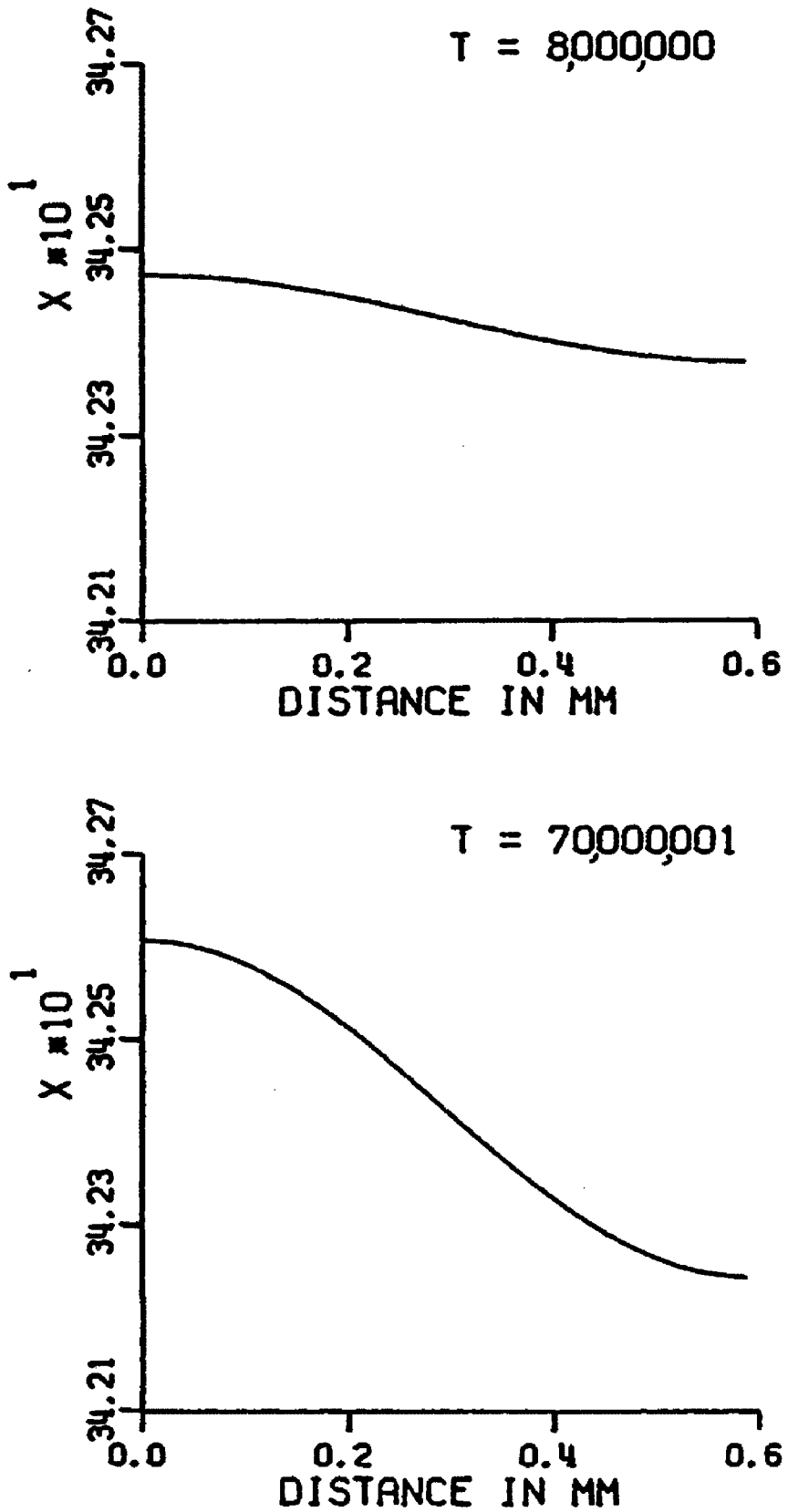
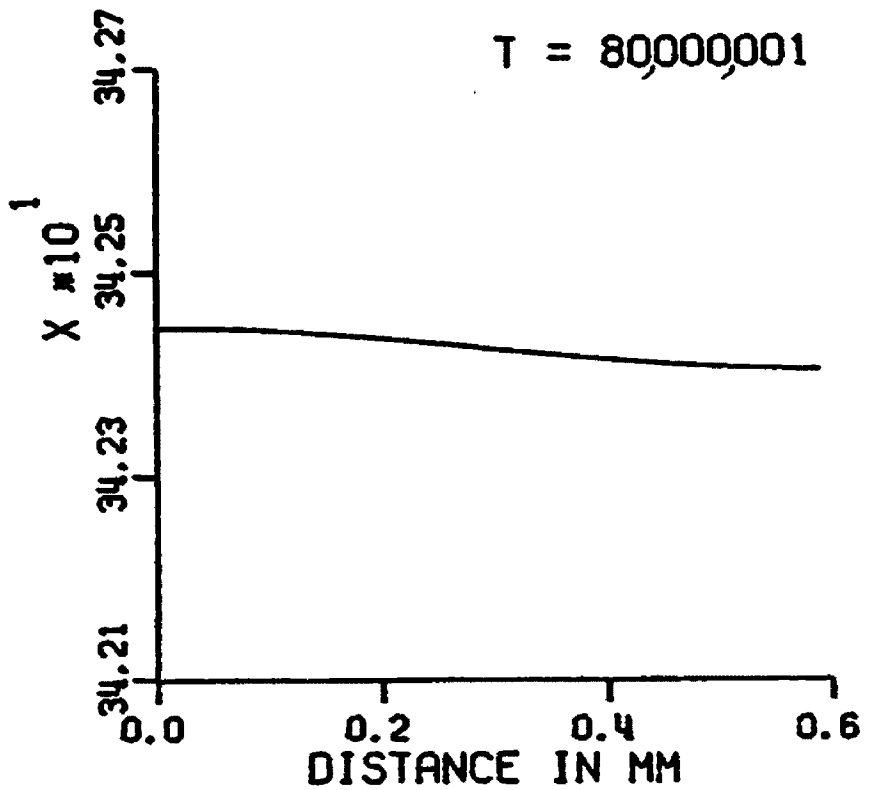


Figure 2.2: Time evolution of first-order solution (x only)
f=1.82 $C=1 \times 10^{-7}$ L=0.5872



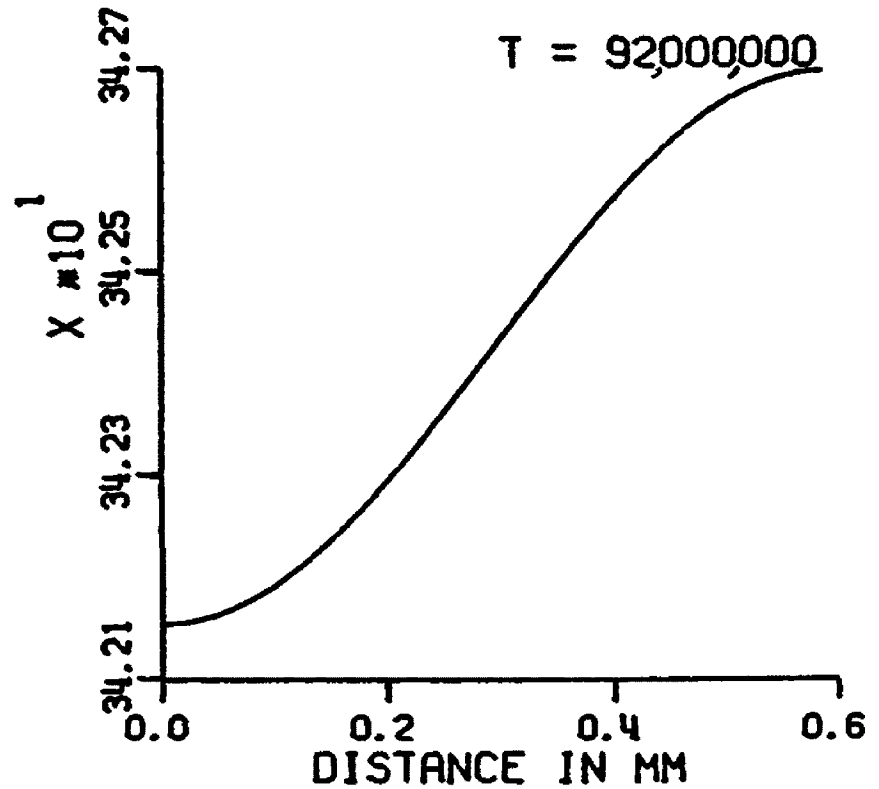


Figure 2.2 (continued)

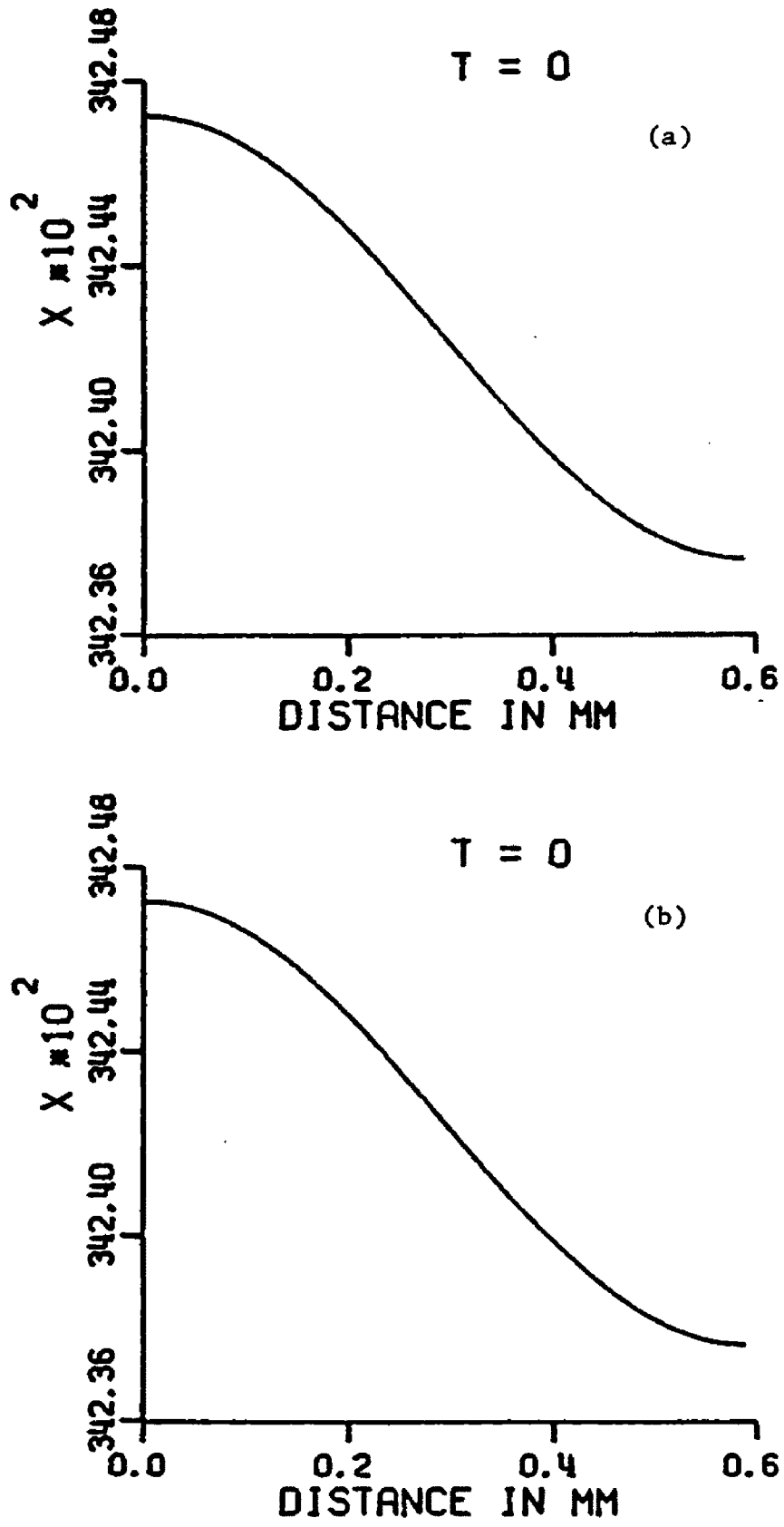


Figure 2.3: (a) First-order solution. (b) 2nd-order solution
f=1.82 $C=4.8 \times 10^{-4}$ L=0.5872

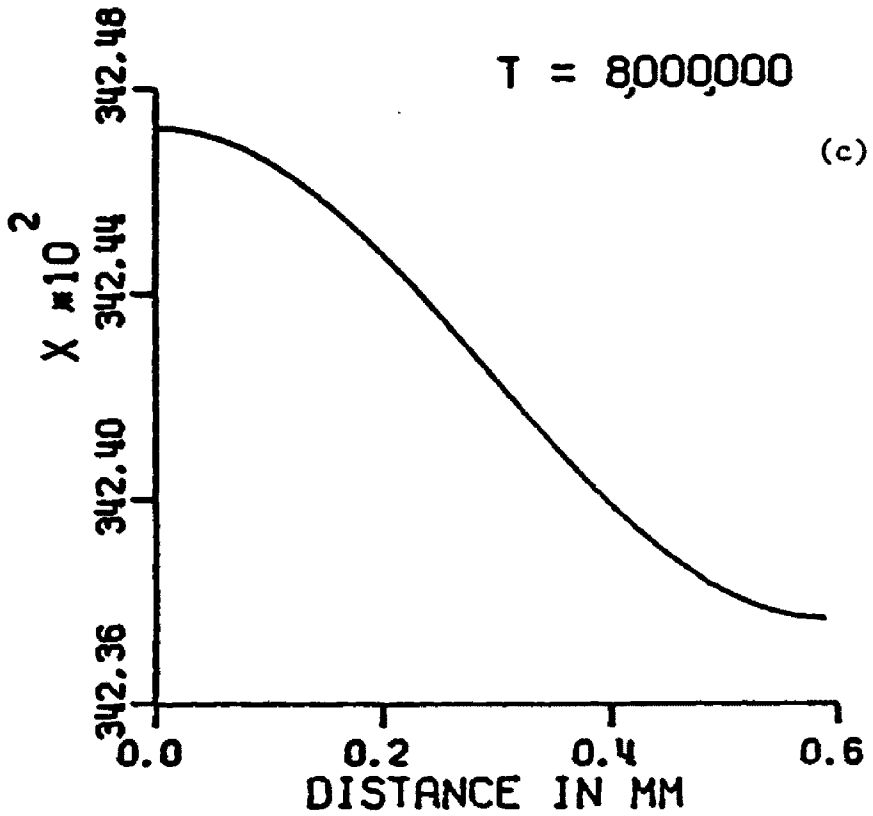


Figure 2.3: (c) Curve evolved from first-order solution begun with $C=1 \times 10^{-7}$.

L = 0.487	Δt = 125	time units
L = 0.687	Δt = 3000	"
L = 0.600	Δt = 500,000	"
L = 0.58721093	Δt = 90,000,000	"

L=0.58721093 mm is the critical length associated with the 5.3500240 value for g ($L=\pi/g$). Figure 2.2 shows the evolution of an x waveform with L critical. Because the clock for Δt begins only when the amplitude of the x wave reaches roughly 0.001, it makes an immense difference in Δt whether the initial perturbation is set with $C=4.8\times10^{-4}$, in which case the t clock starts immediately, or is set with $C=1\times10^{-7}$. Figure 2.3 contains graphs of the initial condition for x with $C=4.8\times10^{-4}$ for the first-order and second-order solutions (see next section for a discussion of second-order solutions), and the graph for the $C=10^{-7}$ perturbation after eight million time steps, the starting point for Δt in this case. The three graphs look virtually identical, yet Δt for the first situation is 785, Δt for the second is 792 and Δt for the third, or small perturbation case, is 90,000,000. This difference explains why most integrations were run with perturbations having $C=10^{-7}$.

If a small initial perturbation has the wrong shape, the waveforms still last a reasonably long time in a container of the proper length. For $C=10^{-7}$ and $x_a=0$ with $y_a=z_a=1$, Δt is still about 20,000. The length of the container is the most important factor in the lifespan of a waveform, followed by the size of the perturbation and then by the shape

of the perturbation. Small perturbations of the wrong shape tend to evolve into wave forms very close to either the first-order solution (with n=1) or a version of that solution multiplied by -1 (i.e., flipped).

The final graph in Fig. 2.2 shows a flip in the x curve at $t=92\times10^6$. This flip usually occurs just before a solution flies off to large values. It is worth noting that the same flip occurs in the Oregonator with zero diffusion. The flip is thus more indicative of chemical concentrations oscillating around steady state values than it is of a diffusion effect. Characteristically, soon after such a flip is seen, the value of x flies up to around 1000, the concentration curves flatten out, and the whole system proceeds to oscillate in unison, a flat solution moving up and down.

3. Other Parameters and Initial Conditions

At one point it was suspected that a small-amplitude first-order solution might stabilize if the value of f was in the stable steady-state region of the Oregonator without diffusion. For the "Low" parameters this region is defined by $f<0.5024$ or $f>2.3545$. Unfortunately, however, small-scale solutions exist only when $0.61<f<1.82$. The best way to increase the f range for solutions without changing reaction rate parameters is to increase the diffusion coefficient for z. Raising D_z artificially high to 0.1 increases the maximum f for a first-order solution to 2.3 -- not quite high enough. Only by switching temporarily to the Field-Noyes[3] parameters is it

possible to find a first-order solution with the value of f in the stable region. The Oregonator using FN parameters has a stable steady state when f exceeds 1.52; with $D_z=7x10^{-5}$ a first-order solution exists for f=1.56. This first-order solution grows very slowly and eventually goes large, just as in the previous example. Setting f greater than 1.52 does not result in stability.

Considering again the "Low" parameters with f=1.82 and equal diffusion coefficients, initial conditions proportional to $\cos(4g\ell)$ instead of $\cos(g\ell)$ were given to the integrator. These are also first-order solutions. For $C=10^{-3}$, the amplitude of the $\cos(4g\ell)$ wave shrank and quickly evolved toward the $\cos(g\ell)$ shape. This behavior proves to be important (see Chapter IV). Waves based on the other eigenvalue for f=1.82, g =28.622757, were tried. Behavior in this case is not qualitatively different from that of the first eigenvalue, although Δt was an order of magnitude smaller. For f=1.82 with D_z increased from $1x10^{-5}$ to $3x10^{-5}$, Δt decreased to 60,000. For f=2.3 and $D_z=10{,}000x10^{-5}$, Δt was only about 30.

Figure 2.4 is a graph of a container of width 4L initialized with four half-wave, first-order solutions. C is set at 10^{-7}. Although the wave kept the shape shown in the figure for a long time at small amplitudes, it added another maximum and minimum as it grew larger and became very unstable before reaching the level at which the Δt clock is normally started. This appearance of a few more extrema than expected is common when starting with integer multiples of a critical length.

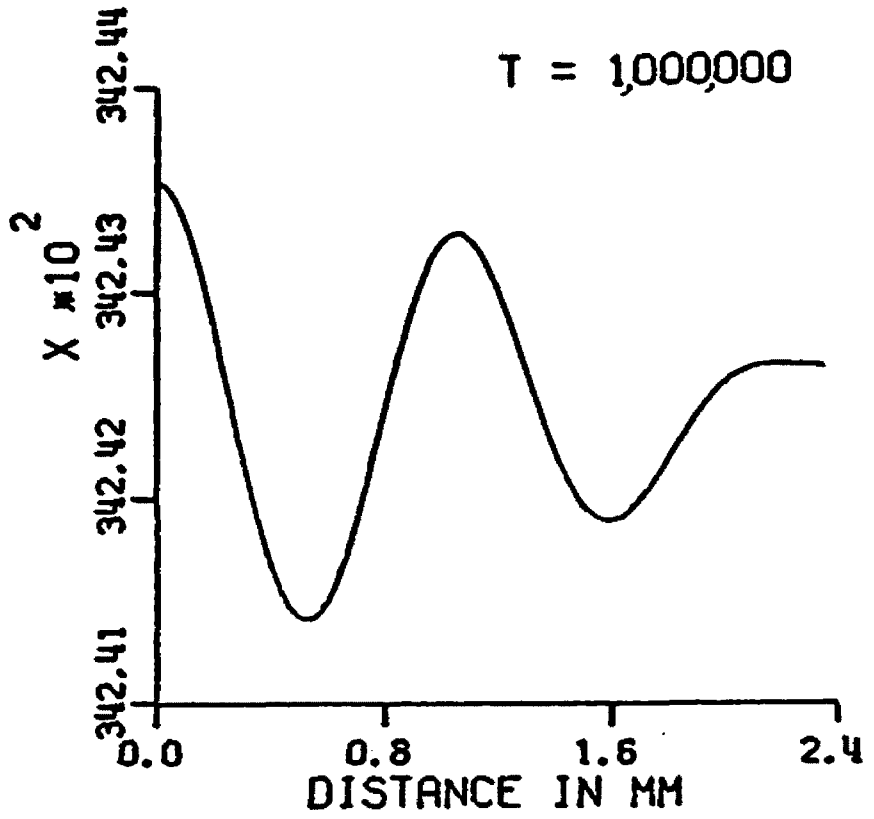


Figure 2.4: Curve evolved from a two-cycle first-order solution.
f=1.82 C=1 x 10^{-7} L=4 x 0.5872 = 2.3488

E. SECOND-ORDER SOLUTION

In the development of first-order solutions, definitions were made so that $x_s = x_0 + x_1 + x^{**}$ where x_0 is the zeroth-order solution, x_1 is the first-order term and x_s the assumed x solution to the non-linear equations. Following through with the variable y again as an example, the following equation may be obtained by substituting $y_0 + y_1 + y^{**}$ for y_s in Eq. 2.2.

$$D_y'(\partial^2 y_0/\partial \ell^2) + D_y'(\partial^2 y_1/\partial \ell^2) + D_y'(\partial^2 y^{**}/\partial \ell^2)$$

$$= [(y_0+y_1+y^{**}) - f(z_0+z_1+z^{**}) + (x_0+x_1+x^{**})(y_0+y_1+y^{**})]/s$$

A number of terms drop out of this equation, as y_0 and y_1 satisfy the zeroth and first-order differential equations. Now, imagining a ** term to have a subscript value of 2, throw away any term on the right whose subscripts total three or more. Since $y^{**} << y_1 << y_0$, these terms are assumed to be small compared to the remaining ones. Calling the variables for this second-order equation x_2, y_2, and z_2, then

$$x^{**} = x_2 + x^{***} \qquad y^{**} = y_2 + y^{***} \qquad z^{**} = z_2 + z^{***}$$

where

$$x^{***} << x_2 \qquad y^{***} << y_2 \qquad z^{***} << z_2 \quad .$$

The three second-order equations are

$$D_x'(\partial^2 x_2/\partial \ell^2) = s[-x_2 - y_2 + x_0 y_2 + x_2 y_0 + x_1 y_1 + 2q x_0 x_2 + q x_1^2]$$

$$D_y'(\partial^2 y_2/\partial \ell^2) = [y_2 - f z_2 + x_0 y_2 + y_0 x_2 + x_1 y_1]/s \qquad \text{Eq. 2.9}$$

$$D_z'(\partial^2 z_2/\partial \ell^2) = w[z_2 - x_2] \quad .$$

Because the zeroth and first-order solutions satisfied the boundary conditions of zero slope at the ends of the container and because the second-order solutions equal $x_0+x_1+x_2$, etc., the second-order term must also have zero slope at the ends. Inserting the explicit first-order solution, noting that $\cos(g\ell)=1/2[1+\cos(2g\ell)]$, and assuming a form for x_2, y_2 and z_2 of

$$\begin{aligned} x_2 &= A_x + B_x \cos(2g\ell) \\ y_2 &= A_y + B_y \cos(2g\ell) \qquad \text{Eq. 2.10} \\ z_2 &= A_z + B_z \cos(2g\ell) \end{aligned}$$

the three equations with boundary conditions become

$$(4g^2 B_x D_x')/s = (1 - y_0 - 2q x_0) B_x + (1 - x_0) B_y - (x_a y_a + q x_a^2)/2$$

$$4g^2 B_y D_y' s = -y_0 B_x - (1 + x_0) B_y + f B_z - (x_a y_a)/2$$

$$(4g^2 B_z D_z')/w = B_x - B_z \quad .$$

The three equations for A_x, A_y and A_z look like the three equations just presented but with all B_i's set to A_i's and g set to zero. The program INS.COS solves these equations. Clearly the second-order terms, being of the form expressed in Eq. 2.10, satisfy the boundary conditions.

As might be expected, the magnitude of the second-order terms is approximately the square of the magnitude of the first-order terms. When, as is most effective, the integrator is given an initial first-order term of order 10^{-7}, the second-order term is of order 10^{-14}. Even in GEARB calculations such extreme precision is lost. When $C=4.8\times10^{-4}$ is used to set the initial conditions, Δt for the first-order solution is 785 while that for the second-order solution is 792. As mentioned earlier, Δt for a solution which evolves from a $C=10^{-7}$ first-order perturbation is approximately 90 million. The addition of a second-order term to a first-order solution is not effective in increasing the longevity of waveforms. Figure 2.5 shows the second-order solution when C=1. It is only at this level of perturbation that the second-order term becomes obvious to the eye.

F. CONCLUSION

For a critical length and a slight perturbation, the GEARB integrator gives x, y and z waveforms looking like half wavelength cosine waves whose amplitudes are related to the values of the eigenvector chosen. These quasi-stable waveforms in some cases grow slowly over millions of time units, or several years in real time. The larger their amplitudes become, the faster these waves grow. By the time they reach the realm where they might be experimentally observable, they are quite volatile, quickly moving to large-scale forms. The lifespans of these small-scale waves are most sensitive to the size of the container. They also respond to the size of the perturbation which

initiates them, and to some extent to the shape of that perturbation.

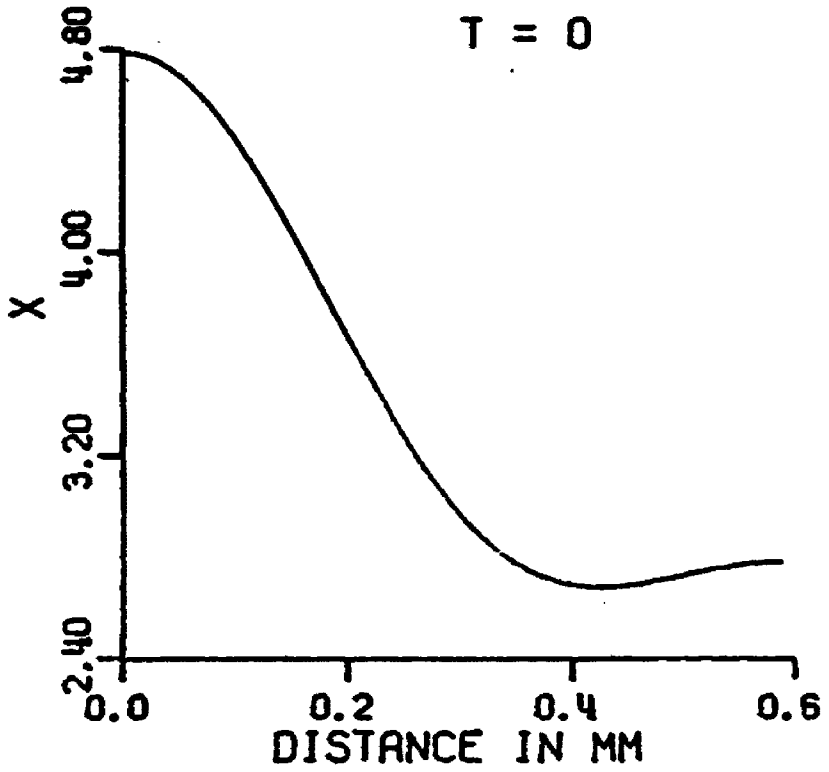


Figure 2.5: Second-order solution.
f=1.82 C=1.0 L=0.5872

CHAPTER III

LARGE-AMPLITUDE, TIME-INDEPENDENT SOLUTIONS

A. PHASE-SPACE ANALYSIS OF THE BASIC OREGONATOR

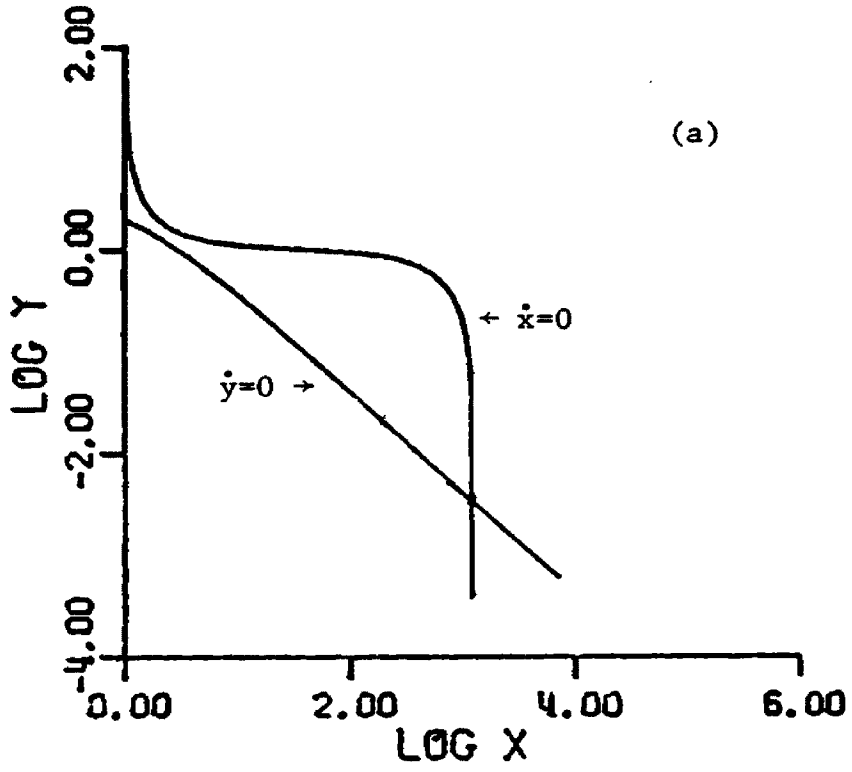
It is valuable before considering the Oregonator coupled with diffusion to present a brief analysis of why the basic Oregonator (without diffusion) oscillates for appropriate values of f. The kinetic equations are

$$(dx/dt) = s(y - xy + x - qx^2)$$
$$(dy/dt) = (-y - xy + fz)/s \qquad \text{Eq. 2.11}$$
$$(dz/dt) = w(x - z) \quad .$$

They represent the changes with time in bulk chemical concentrations in a well-stirred container. One could look at the three dimensional phase space for x, y and z, but since z couples fairly simply with the x and y equations, and since z changes relatively slowly compared to x and y (w is considerably less than s or 1/s), it is simpler to look at the two dimensional x-y phase space. First assume that z is a constant and then let z change.

Figure 3.1 shows in the x-y plane the isoclines, or lines defined by letting each time derivative equal zero. Setting $\dot{y}=0$,

$$y = fz/(1 + x) \quad .$$



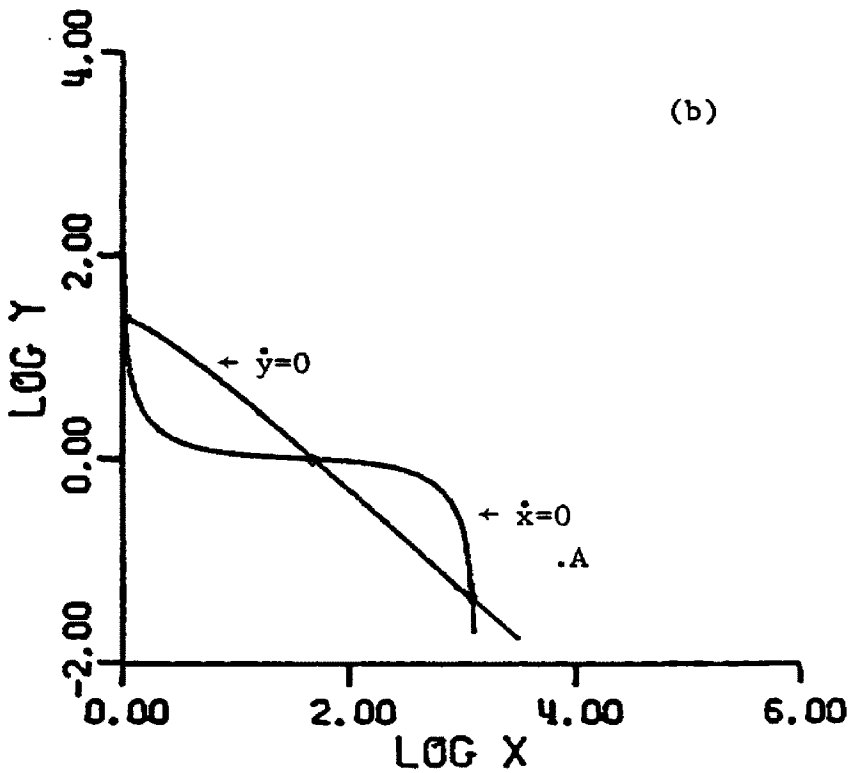


Figure 3.1: Isoclines in phase space over time.
"Low" parameters f=1
(a) z=4 (b) z=49.5 (c) z=350

(c)

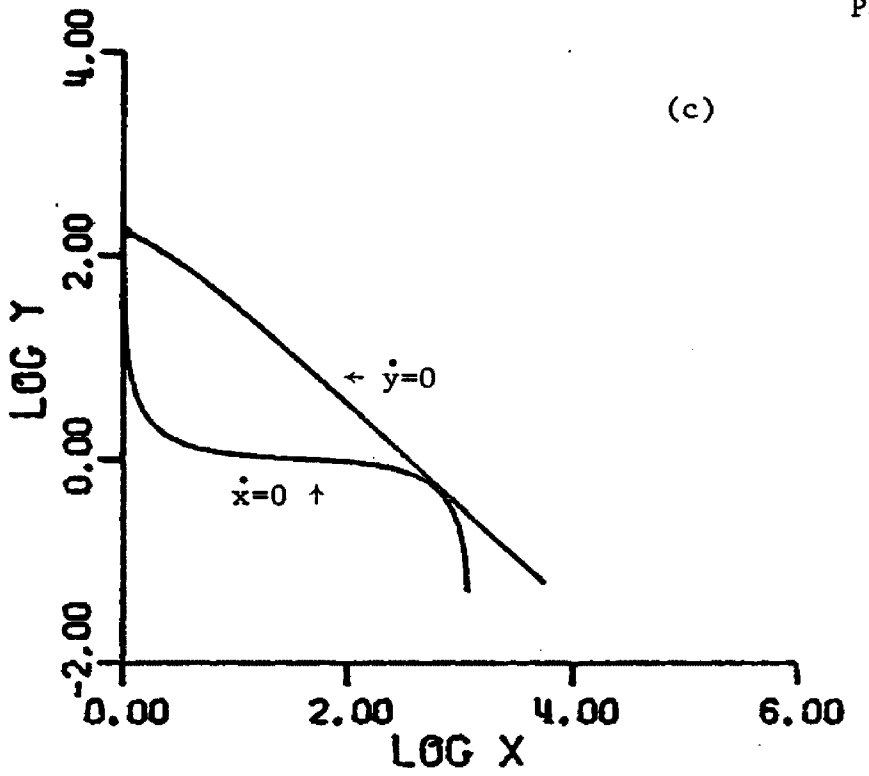


Figure 3.1 (continued)

Setting $\dot{x}=0$ gives

$$y = (x - qx^2)/(x - 1) \quad .$$

The $\dot{y}=0$ isocline is a hyperbola with asymptotes at $x=-1$ and $y=0$. Only mathematical behavior in the first quadrant is of interest as concentrations must be positive. The $\dot{y}=0$ isocline opens upward in this region, and the $\dot{x}=0$ isocline is monotonic decreasing from its asymptote at $x=1$. The latter isocline is negative in the region $0<x<1$, so this portion of the isocline is not of interest. The $\dot{x}=0$ isocline has one inflection point at $x=(1-q)/2q$ and crosses the abscissa at $x=1/q$. Note that the $\dot{x}=0$ isocline is independent of z. Figure 3.1b shows the isoclines when $z=49.5$, the steady state z value for the Oregonator system when $f=1$. Log scales on the axes help clarify the intersections of the two curves and also make the effect of raising or lowering the value of z be roughly to raise or lower the $\dot{y}=0$ isocline. See Figs. 3.1a,b,c. In a guest lecture at the University of Montana, Professor William C. Troy of the University of Pittsburgh pointed out the value of these diagrams as a tool for understanding why the Oregonator oscillates.

Figure 3.2 is an enlarged version of isoclines with $z=49.5$. If a trajectory starts at any point in this phase space and it is assumed temporarily that z remains constant, the trajectory will extend through the space as time evolves. If the trajectory moves across the $\dot{x}=0$ isocline, it must be moving in a vertical direction, as $dx/dt=0$. Similarly, if a path moves across the $\dot{y}=0$ isocline, it must move horizontally. Vertical and horizontal lines are drawn on the isoclines

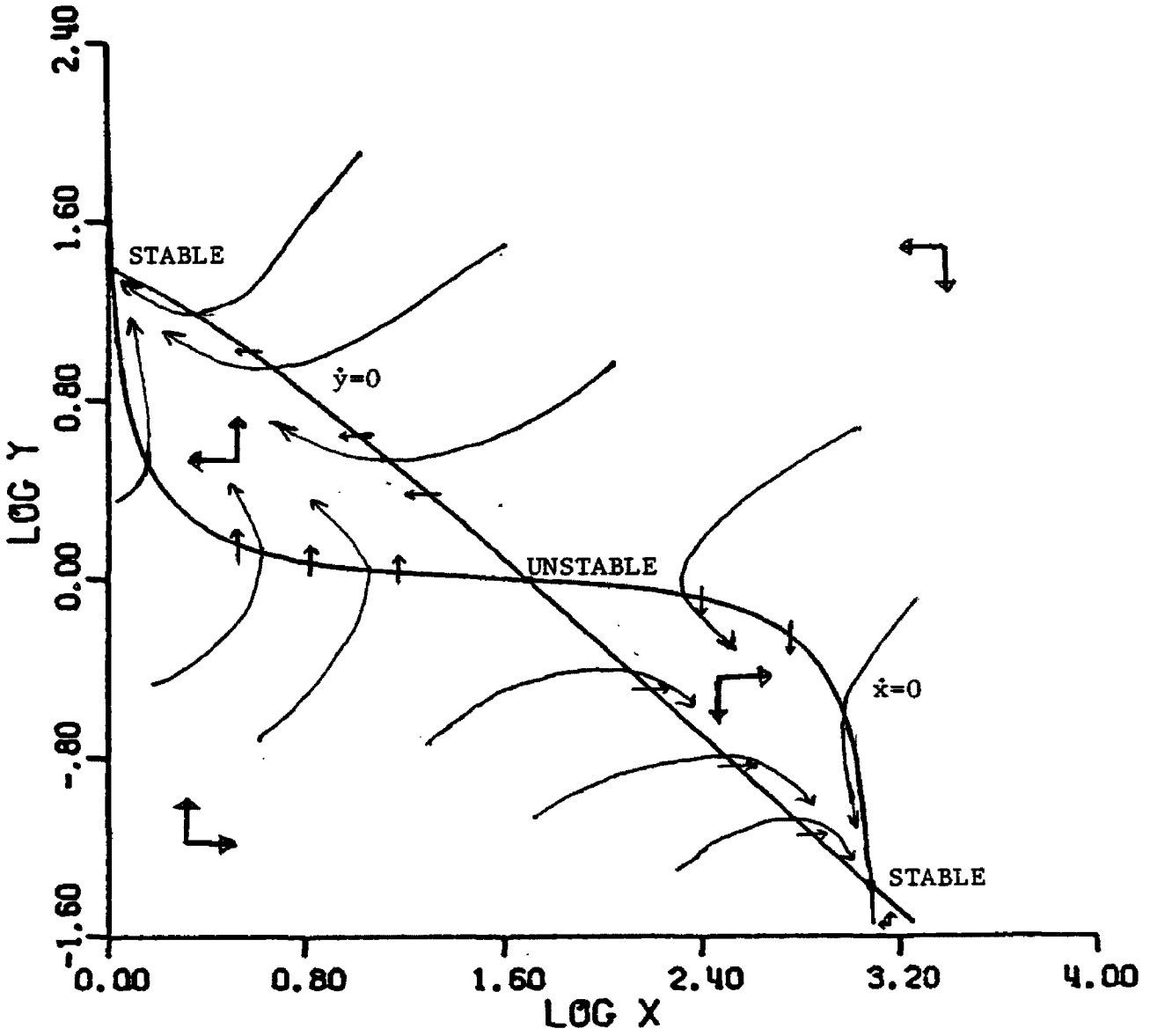


Figure 3.2: Isoclines in phase space over time with trajectories sketched in (z=49.5).

in Fig. 3.2 as a reminder of this behavior.

Looking at the signs of $\dot{x}$ and $\dot{y}$ on either side of the isoclines, it is clear that if both x and y start large, they will both get smaller, for in that case the x^2 and xy terms will dominate the equations. This general direction of motion (down and to the left) is indicated by the double arrows in the upper right corner of Fig. 3.2. A trajectory moving down will continue to do so until it crosses the $\dot{y}$=0 isocline. On the other side of that isocline it will always move up. This is true because, very near the y isocline, x is monotonic and dy/dt varies linearly with x. A similar switch in horizontal direction will occur as the trajectory crosses the $\dot{x}$=0 isocline. Using this information it is not difficult to draw double arrows indicating trajectory directions in each major region of phase space. Remembering that a trajectory only moves horizontally or vertically when on one of the isoclines, it is then possible to sketch in the rough character of the trajectories for these equations, as has been done in Fig. 3.2. In this case there are two stable nodes and a central unstable node. All trajectories evolve toward the stable nodes.

The above analysis was made assuming that z is constant. Now consider what happens when z begins to move. Remember that x and y move more quickly in time than does z. Suppose the Oregonator starts at point A in Fig. 3.1b with z set initially at 49.5. The trajectory will quickly move close to the right-hand equilibrium point and x will approach 1000. However dz/dt=w(x-z)=w(1000-49.5), so z will gradually

get larger, raising the $\dot{y}$=0 isocline. The trajectory will follow the slowly-moving, attractive equilibrium point until the $\dot{y}$=0 isocline reaches a position like that in Fig. 3.1c. At that time the nature of the trajectories has changed. The right-hand stable node and the unstable node have come together and disappeared; suddenly all trajectories are drawn to the one remaining attractive node on the left. The trajectory being followed will quickly move near the left-hand node. The value of z had been pulled up to about 350 before the disappearance of the two nodes. Now, however, x=1 so dz/dt=w(1-350), a negative number. The value of z decreases and the $\dot{y}$=0 isocline drops. The trajectory this time follows the left equilibrium point until the situation pictured in Fig. 3.1a occurs, at which time the trajectory moves to the right again. This cycle continues, creating continuous periodic oscillations in the Oregonator.

B. WIND INTERPRETATION OF PHASE SPACE

Now consider the Oregonator with diffusion. The equations for time-independent solutions are

$$-D_x'(d^2x/d\ell^2) = s(y - xy + x - qx^2)$$

$$-D_y'(d^2y/d\ell^2) = (-y - xy + fz)/s \qquad \text{Eq. 2.12}$$

$$-D_z'(d^2z/d\ell^2) = w(x - z) \ .$$

The right-hand sides of these equations are the same as those in the previous analysis. Figure 3.3 shows the curves in phase space obtained by assuming z constant and setting successively the second derivative of

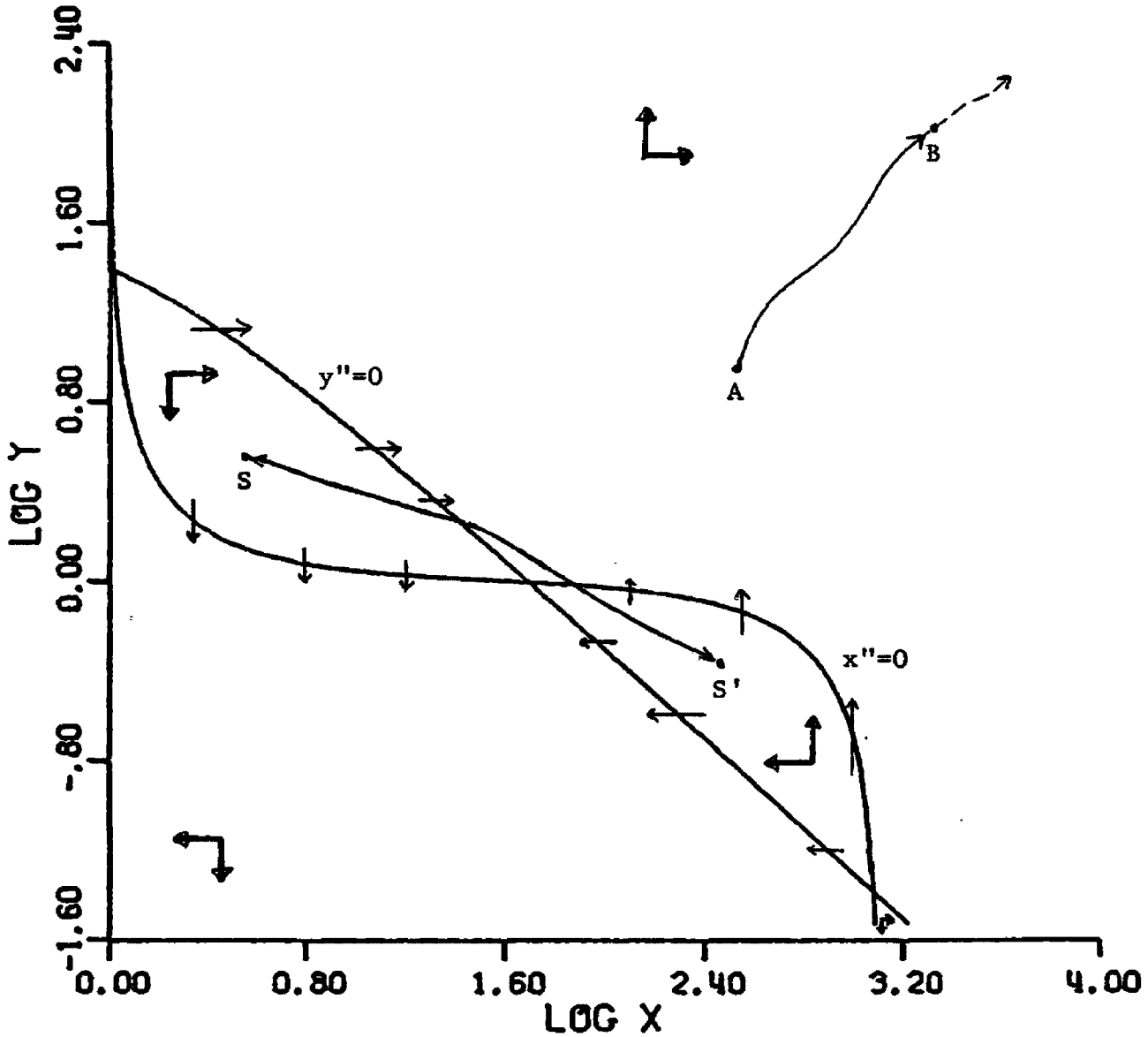


Figure 3.3: "Trade" lines in phase space over distance. Path S-S' could satisfy boundary conditions. Path A-B could not.

x and y to zero. These curves are the same as the isoclines drawn in Fig. 3.2, but since a second derivative, not a first, is set to zero, they are no longer isoclines. Moreover, now distance moves a trajectory through phase space when the container is scanned from one side to the other. Since containers are finite, the lines in phase space must also be finite.

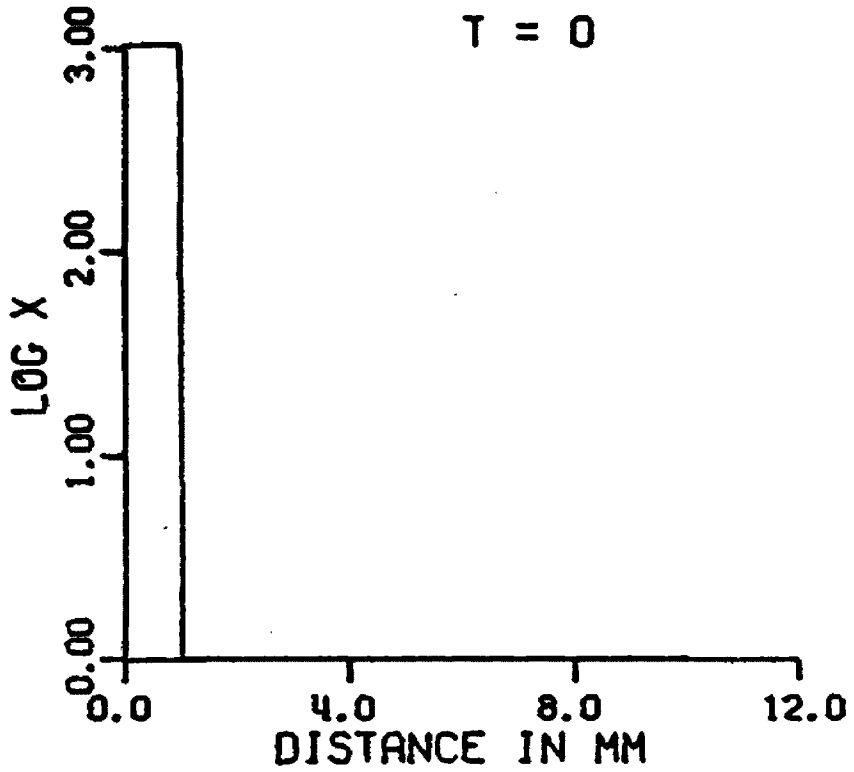
In developing a feeling for which trajectories can satisfy the given boundary conditions, it is helpful to consider temporarily the independent variable distance as a pseudo-time. This change in viewpoint should not be confusing since the solutions to the Oregonator considered here are independent of real time. The derivative $dx/d\ell$ then is interpreted as a velocity with units mm^{-1} and the second spatial derivative becomes an acceleration with units mm^{-2}. Using pseudo-time, a trajectory crossing the $x''=0$ line then has a zero acceleration, or constant velocity, in the horizontal direction. A parallel statement is true for the $y''=0$ line. Eq. 2.12 can now be interpreted as an equation of motion for a kind of force whose magnitude and direction at any point in phase space is given by the expressions on the right sides of Eq. 2.12. This "force" has dimensions of mm^{-2}. Call this force a chemical wind. Trajectory motion is controlled by these winds. The directions of the possible extremes of the winds in each region of space are indicated in Fig. 3.3 by double arrows. Since the equations differ only by a constant and a minus sign from those in the previous section, the double arrows are exactly reversed from the ones in Fig. 3.2. The short lines drawn on the $x''=0$ and $y''=0$ lines this time indicate the

direction of the wind at those points (not the direction of the trajectory). Points of intersection are points with no wind at all.

Now consider the boundary conditions of $dx/d\ell=dy/d\ell=0$ at the sides of the container. In real space these zero first derivatives mean that concentrations do not change at the walls of the container; in phase space using pseudo-time they mean that a point sweeping out a trajectory must have zero velocity at the beginning and end of its path. The use of this wind artifice, though not helpful in selecting specific trajectories, now makes it possible to see at a glance what types of trajectories possibly can satisfy the boundary conditions. A trajectory starting at point A in Fig. 3.3 would continuously accelerate up and to the right. Always moving with the wind, as it were, there is no way that the moving point could slow down and stop. Yet its velocity must be zero when the far wall is reached. This contradiction indicates that there is no time-independent solution that starts at point A (note again that at this stage z is assumed constant). The only way that a point sweeping out a trajectory can slow and stop is if its motion takes it into a region where it is moving directly into the wind arrows. This motion can only happen if the point moves from one of the regions between the unstable node and a stable node to the other such region. See as an example line S-S' in Fig. 3.3. Changes in z have not yet been taken into account in this discussion, nor has the question of the time stability of such a solution as line S-S' in the face of small, random spatial perturbations been addressed. However, it is in the area just described that stable solutions are found.

C. RESULTS: LARGE-AMPLITUDE SOLUTIONS

Following Ermentrout, Hastings and Troy[14] in their analysis of a two-variable model which has similarities to the three variable situation, investigations in this work were begun by raising D_z artifically high, from $1x10^{-5}$ to $10{,}000x10^{-5}$. Since in a time-independent solution of Eq. 2.12, $d^2z/d\ell^2=w(z-x)/D_z'$, making D_z very large makes the variations in z quite small. Picking f=2.5 and leaving all rate constants and diffusion coefficients except D_z at their normal "Low" values, the program WIND finds that the x" =0 and y" =0 lines intersect in three points only when z is between 2.33 and 125.40. An intermediate value z_k was picked and the GEARB integrator was initialized by setting the z value for all 250 spatial points equal to z_k. WIND also gives, for the chosen value z_k, the x and y values at the three intersection points. Note that these points correspond to the stable and unstable nodes in phase space over time (see Fig. 3.2). Concentrations for x and y were initialized as a square pulse shape, making the pulse height equal to the x and y concentrations at one stable node and the quiescent height equal to the values at the other stable node. See Fig. 3.4. Use of these initial conditions was an educated guess based on the conclusion drawn from phase-space diagrams that any time-independent solution to the Oregonator must have points somewhere in the two closed regions defined by the intersection of the x" =0 and y" =0 lines. The waveforms quickly evolve to the stable shapes shown in Fig. 3.5. Figure 3.6a shows the x-y phase-space diagram for this solution, and Fig. 3.6b shows the x" =0 and y" =0



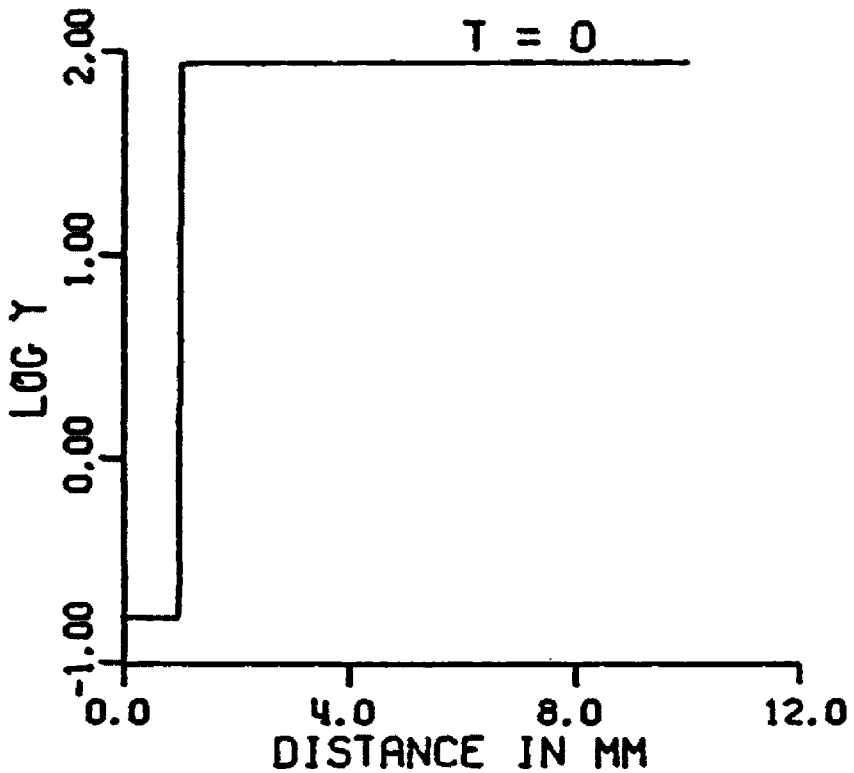


Figure 3.4: Initial pulse with levels set at values of extreme intersections in wind diagram. "Low" parameters. f=2.5 D_z=0.1 $cm^2 sec^{-1}$ L=10

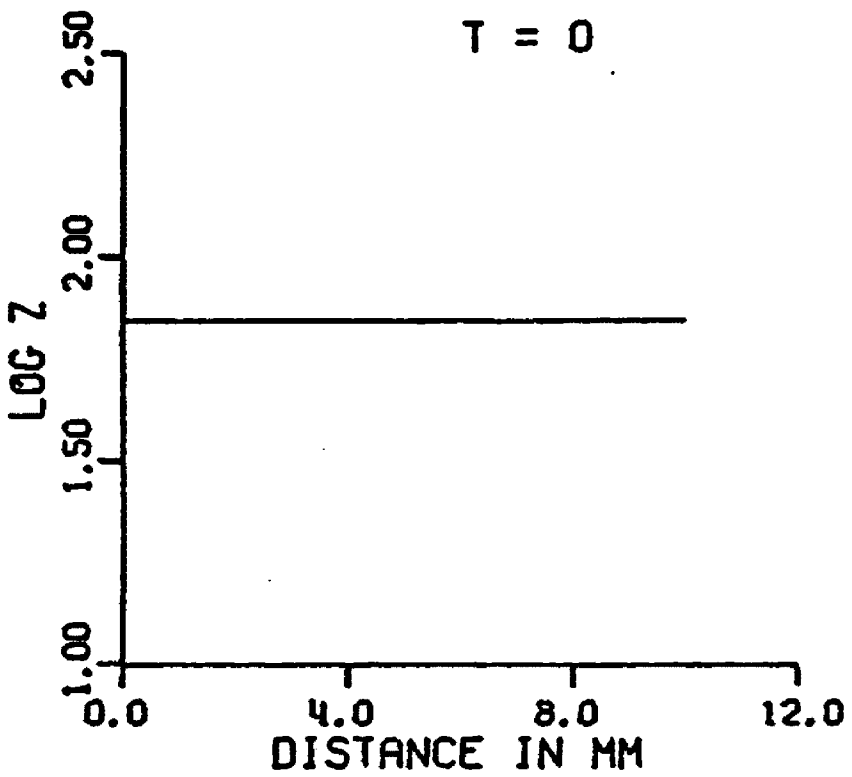


Figure 3.4 (continued)

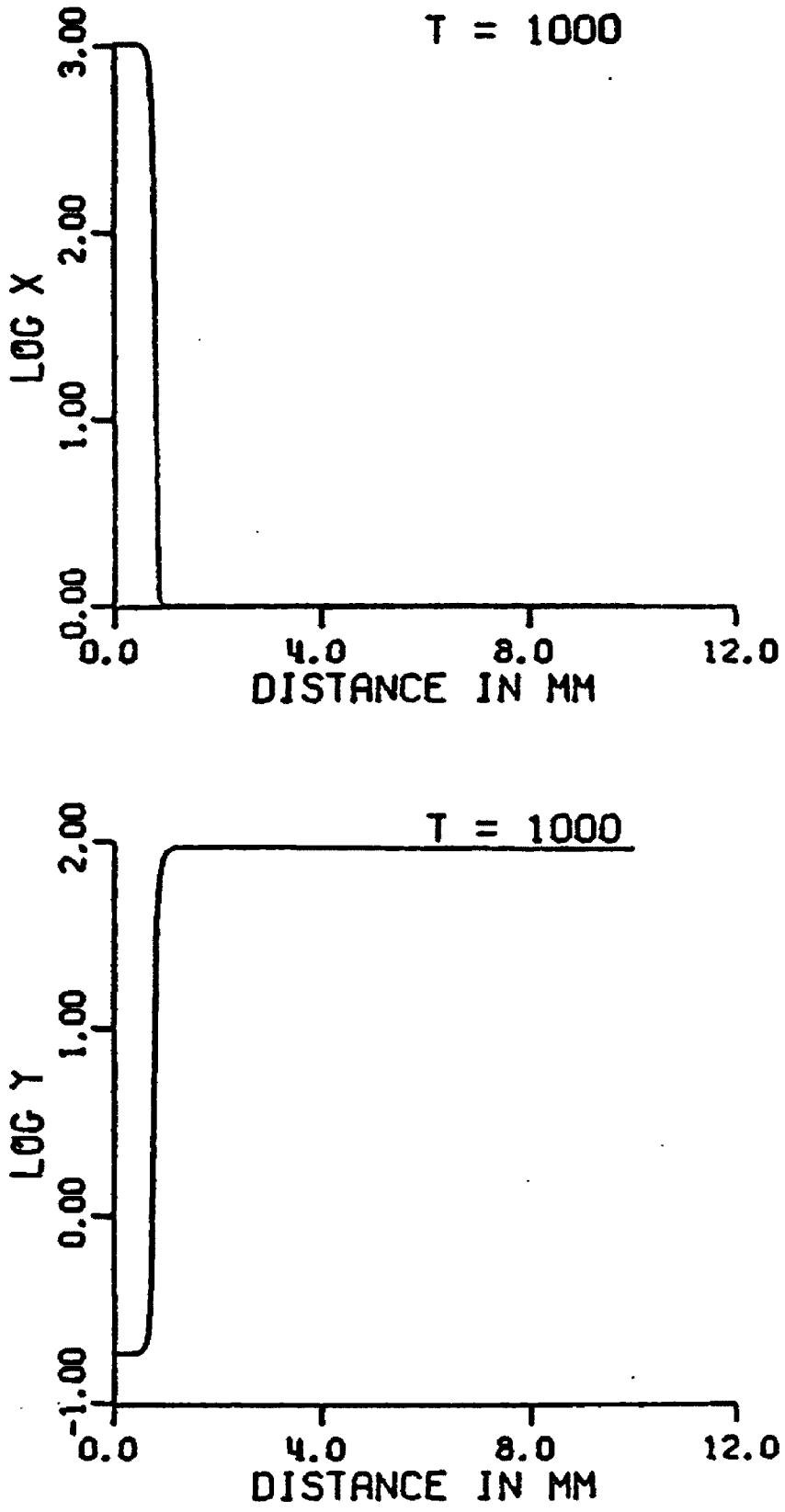


Figure 3.5: Stable solution. "Low" parameters.
f=2.5 D_z=0.1 $cm^2 sec^{-1}$ L=10

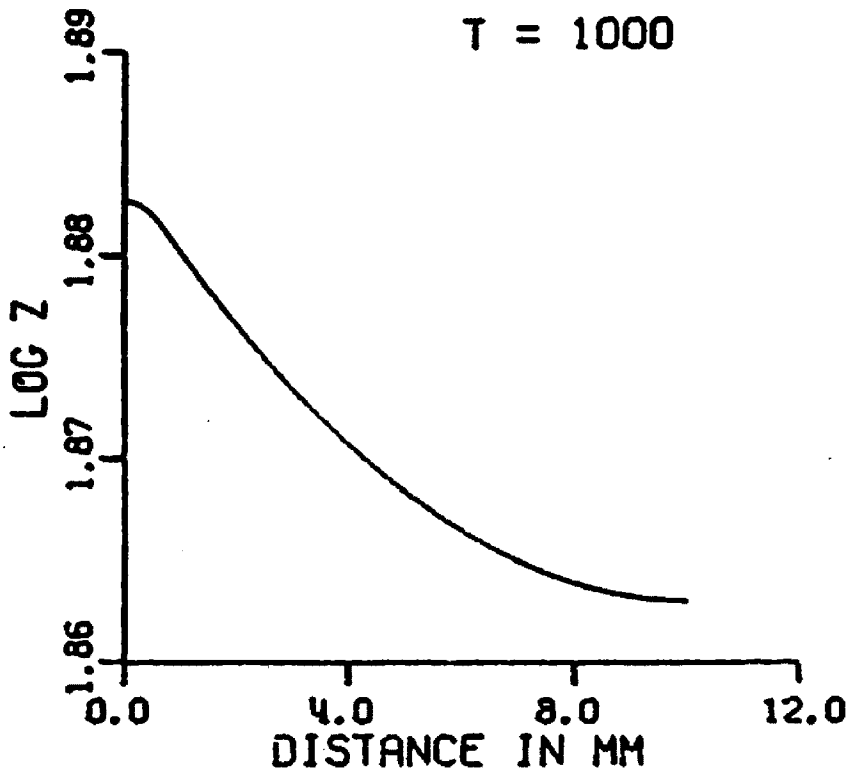


Figure 3.5 (continued)

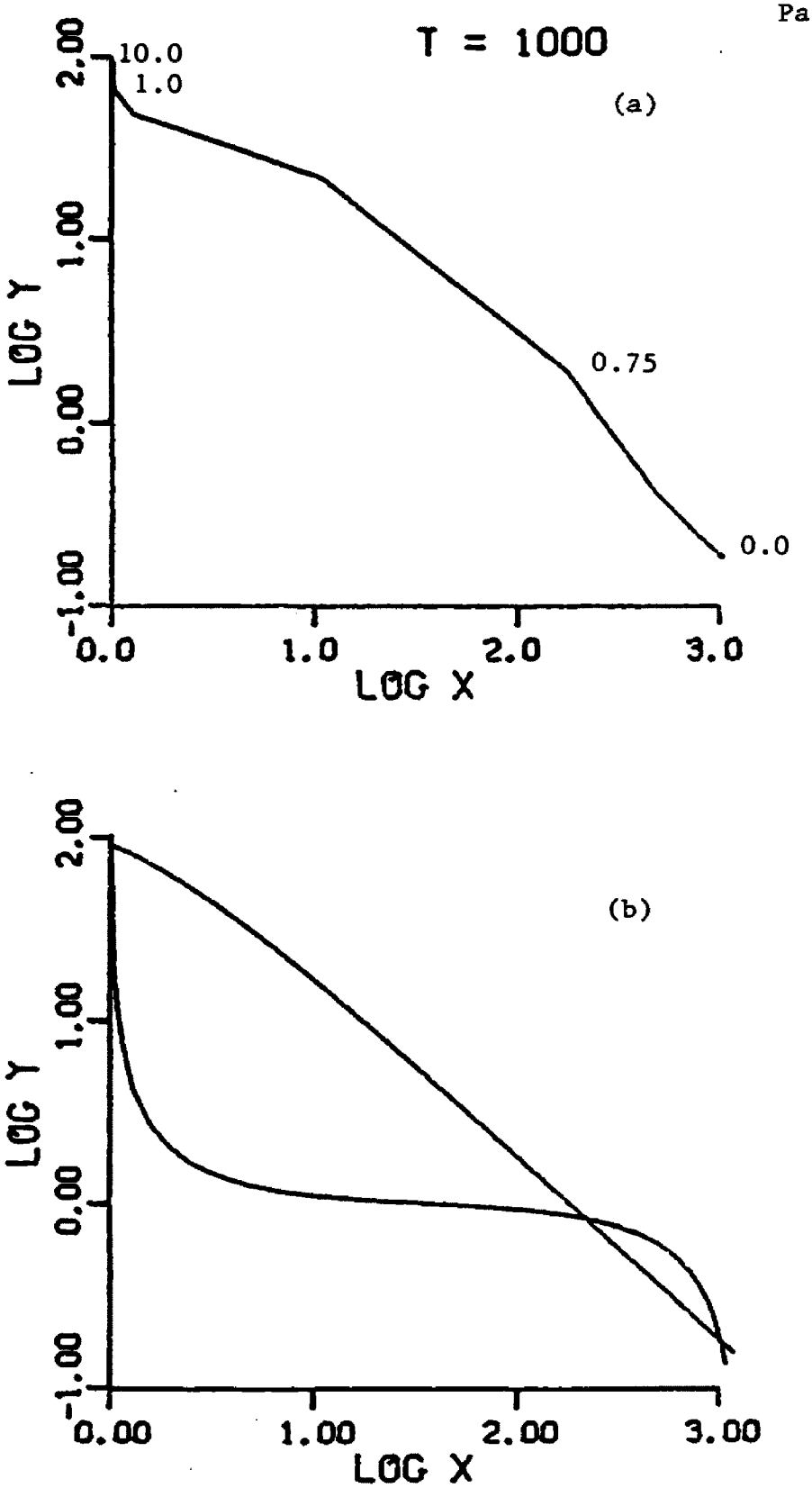
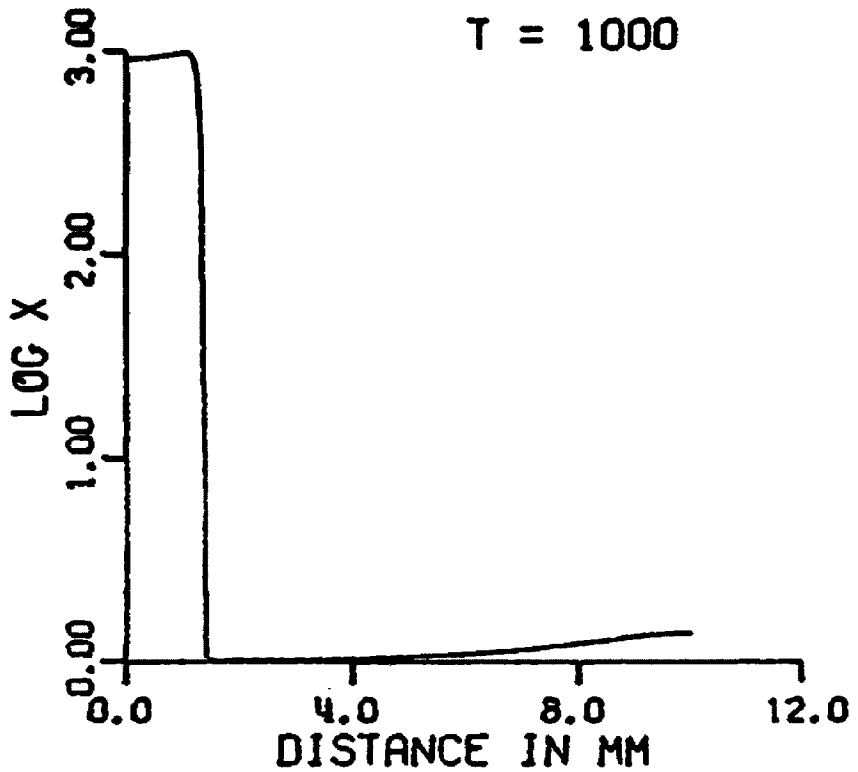


Figure 3.6: (a) Phase-space graph of stable solution in Fig. 3.5. Numbers on path indicate distance in real space. (b) Wind diagram with log(z)=1.863

lines for z_k equal to the minimum z value over all 250 spatial points. Heretofore, the type of lines shown in Fig. 3.6b will be referred to as trade lines, and the diagram as a whole will be called a wind diagram. A wind diagram using z_k set to the maximum value for the z variable in this solution looks virtually identical to the diagram shown for the minimum z, for z changes very little. When D_z is reduced to $1{,}000\times10^{-5}$, a similar stable solution is found, though this time z does change appreciably. For $D_z=100\times10^{-5}$, the waveforms change shape with time and do not stabilize.

If f is reduced to 1.0 and the waveforms initialized as described above (setting an x-y pulse tied to the two extreme intersection points in a wind diagram and using an appropriate guess for z_k), a stable solution is found even when D_z is as low as 100×10^{-5}. Setting f=0.6, $D_z=50\times10^{-5}$ and reinitializing in the usual manner, the stable wave forms shown in Fig. 3.7 are found. Figure 3.8 shows the phase-space diagram for this solution, and also two wind diagrams. Figure 3.8b is drawn with z_k set at its minimum value and Fig. 3.8c is with z_k at its maximum. Note in the earlier solution, shown in Fig. 3.6, that if that wind diagram is laid on top of the phase-space diagram, the solution goes virtually from one stable node to another. Here the phase-space diagram is more complicated, though still useful. Point A in the phase diagram (Fig. 3.8a) corresponds to the left node in the wind diagram for small z_k. As z increases (in real space z is small on the right side of the container, so motion to increase z is actually from right to left in real space), the wind diagram gradually changes from that in



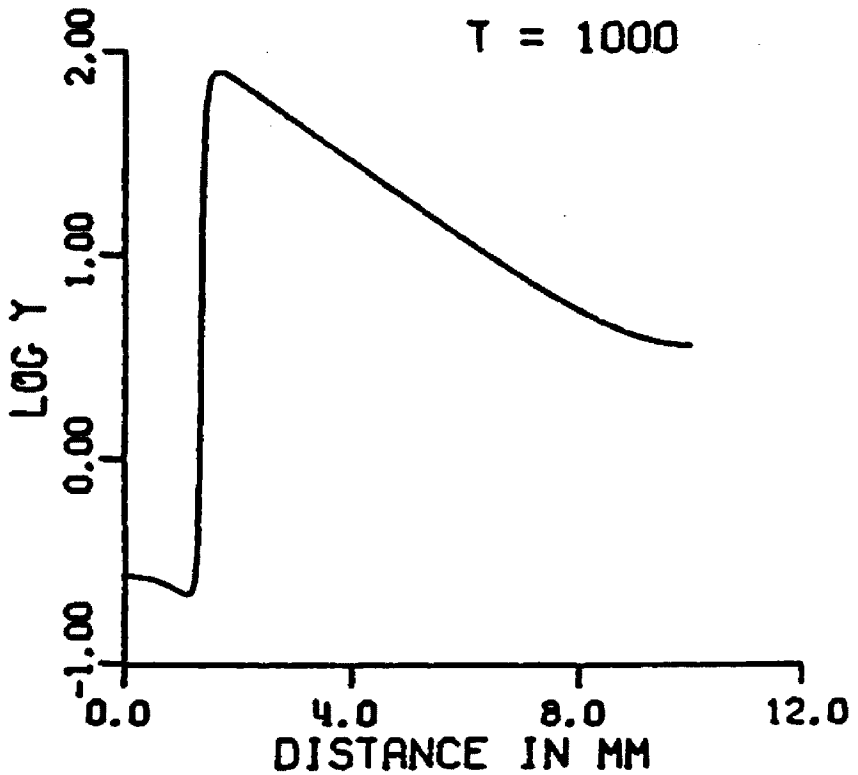


Figure 3.7: Stable solution. 1125 spatial point grid. "Low" parameters. f=0.6 D_z=50 x $10^{-5}cm^2sec^{-1}$ L=10.

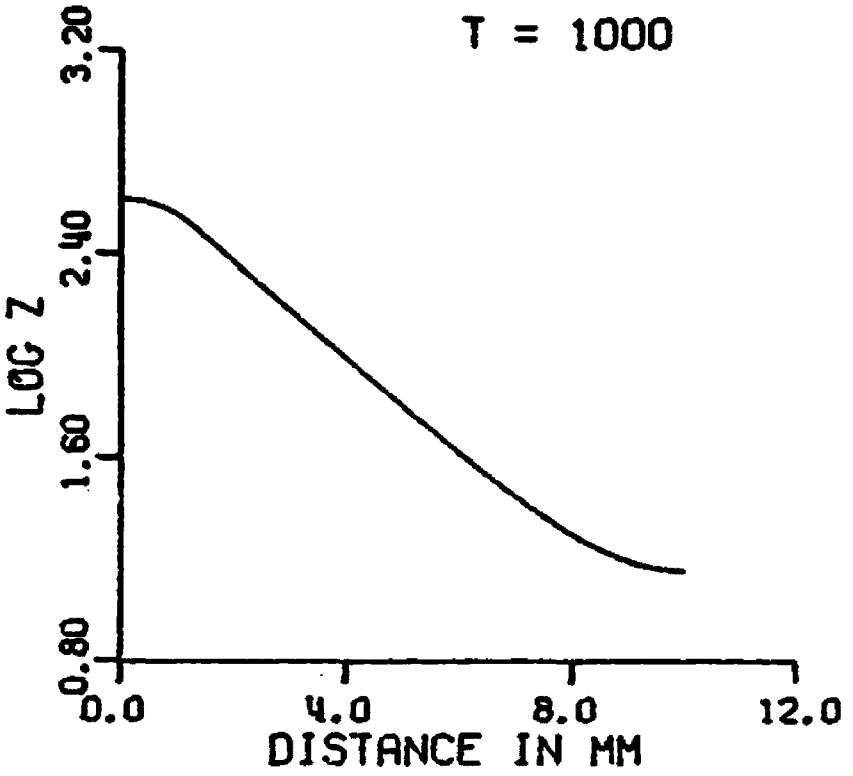


Figure 3.7 (continued)

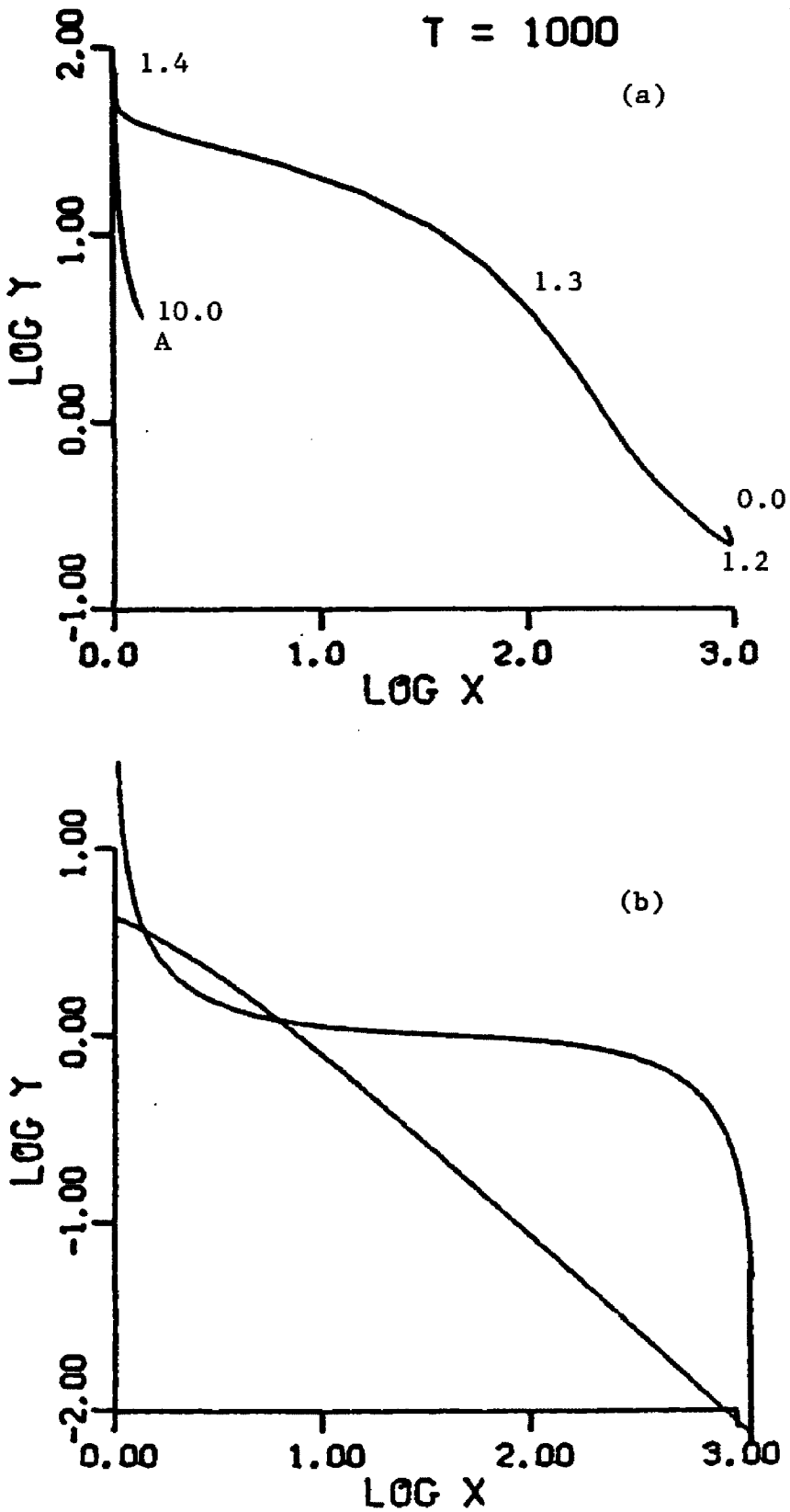


Figure 3.8: (a) Phase-space graph of stable solution in Fig. 3.7
(b) Wind diagram with log(z)=1.153

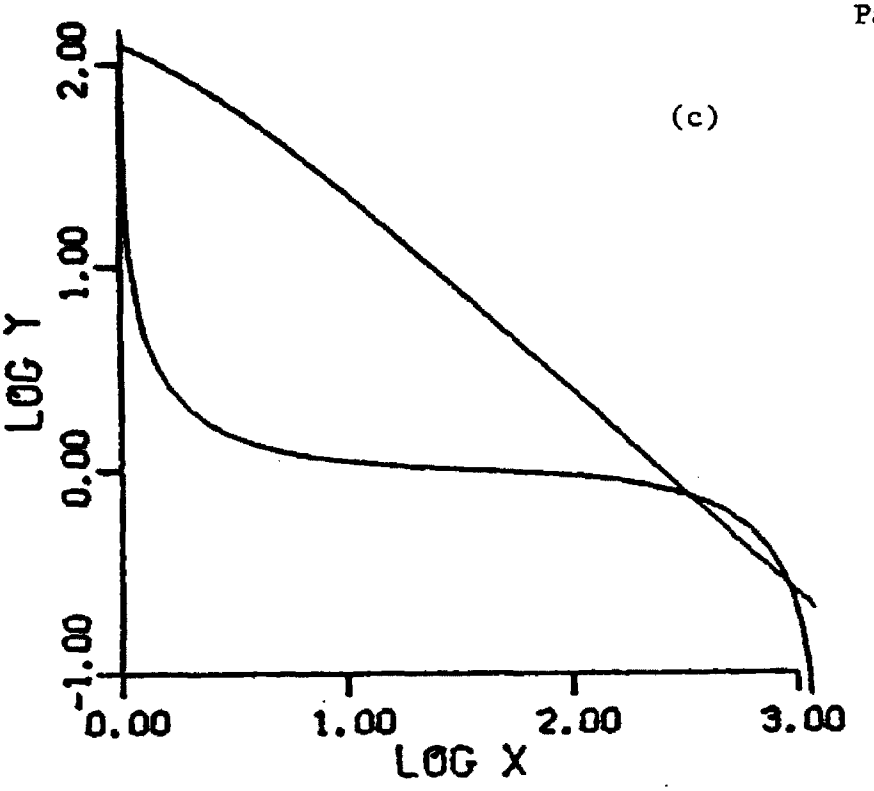


Figure 3.8: (c) Wind diagram with log(z)=2.613

Fig. 3.8b to that in Fig. 3.8c. The phase-space trajectory follows the left stable node as it slowly rises. When the point is reached where x and y change magnitudes, the phase-space point jumps quickly from the left to the right stable node. This jump is better imaged by saying that the solution is stretched between two nodes. The long part of the trajectory actually involves only a relatively few spatial points. The jog in the right-hand end of the phase-space diagram can be seen to follow the slowly moving right-hand node. All stable waveforms which evolved on the GEARB integrator were stretched between the two stable nodes. No other trajectories of those suggested as possible by the wind diagrams ever stabilized.

There are three conditions which seem to be necessary if a given waveform is to be stable. First the y trade line must intersect the x trade line in three points for every value of z in the solution. In one case there seemed to be a violation of this rule: the trade lines intersected in only one point over most of space, yet the x and y waveforms were very narrow stable spikes centered in the middle of a very wide container. Suspecting that such a large narrow spike might be a numerical artifact of GEARB, the number of spatial points for the given container width was increased from 250 to 1125. The spikes were indeed artifacts and disappeared on this finer grid. One of the more typical solutions was also tested on this finer grid and that solution was unchanged (Fig. 3.7).

The second necessary condition for stable waveforms is that all values of z in a solution must be between the high and low values of x. This condition can be seen mathematically. If z starts larger than all values of x, the z curve will initially have a positive curvature since $d^2z/d\ell^2=-w(x-z)/D'_z$. Since z starts with zero slope, this curvature will make z larger. Ever larger values of z force the curvature to remain positive. The inflection point necessary to allow z to have a zero slope at the far wall can never occur. If z starts within the bounds of the x range but grows larger than the largest x, it will have a positive slope as it leaves the x range. Again, positive slope and always larger positive curvature at a point means the z curve can never level off as is necessary when a wall is reached. Similar reasoning holds for the case where z becomes smaller than the minimum x. The value of z must stay within the range of x. A search for contradictions to this rule yielded none.

The third condition is that neither the width of the container nor the width of the initial pulse can be too short. The appropriate pulse and container widths for a set of diffusion coefficients are extrapolated from earlier stable solutions which were found for similar parameters.

The question of which initial waveforms will evolve in time to stable shapes was studied. Initial conditions need not be set precisely. The z values can be set at any reasonable value between the high and low x values. The restrictions on initial x and y values, when

starting with pulse waveshapes, can best be described in reference to Fig. 3.2, where time is the independent variable. Most points in a rectangular pulse have zero curvature. Thus the partial differential equations (Eqs. 1.2-1.4) which describe the initial time evolution of the input pulse are, for most points, virtually identical to the time-dependent ordinary differential equations (Eq. 2.11) for the Oregonator without diffusion. When an initial pulse has points in phase space on either side of the separatrix (a line which divides phase space over time into two attractive regions), each point will quickly move close to one or the other of the nodes, creating a pulsed waveform quite close to a stable waveform. This restriction leaves great latitude in initial conditions.

D. INITIAL SURVEY OF SOLUTIONS

Two parameters, f and D_z, were manipulated in attempts to meet the three conditions presented above. Though changing f moves the position of the y trade line, the relationship of f and D_z to the values of z in a stable solution is not yet clear. Table 3.1 summarizes the experimental findings thus far.

Most of the stable waveforms listed in Table 3.1 look basically like those of Fig. 3.7. It seems that in those waveforms there must be a sufficient distance after the pulse for z to gradually adjust to a line whose slope is zero. Lengthening the container beyond a critical minimum length does not affect the waveshape. Note that since these solutions have zero slopes at the edges, it is possible to place two

f	$D_z * 10^5$	Pulse Width	z values	Allowed z Range	U-Unstable S-Stable
2.5	10,000	0.75 mm	72.9-76.3	2.3-125.4	S
2.5	100	—	z > 125.4	2.3-125.4	U
1.0	100	0.85 mm	21-215	5.8-313	S
0.6	100	1.80 mm	47-404	9.7-522	S
0.6	50	1.30 mm	14-410	9.7-522	S
**0.6	10	1.10 mm	57-422	9.7-522	S
0.6	3	—	z > 522	9.7-522	U
0.6	1	—	z > 522	9.7-522	U
0.1	100	4.25 mm	251-1122	58.1-3135	S
0.1	10	—	z < 58.1	58.1-3135	U
0.01	10	—	z < 581	581-31,350	U

Table 3.1: Summary of stability results for a range of values for f and D_z.

such solutions side by side. Making use of these periodic boundary conditions it is possible to construct, for example, solutions having one pulse (double the normal width) centered in a container, or single pulses at each edge of a container which is twice the normal minimum width (see Fig. 4.3).

There is one stable waveform, the one marked ** in Table 3.1 and exhibited in Fig. 4.1, which seems to have a different character than the others. This waveform provides a link between small-scale quasi-stable solutions and large-scale stable solutions. It will be discussed in the next chapter.

CHAPTER IV

CONNECTIONS AND SPECULATIONS: WORK IN PROGRESS

A. CONNECTIONS

The special stable waveform alluded to at the end of the previous chapter has an f value of 0.6 and D_z equal to $10x10^{-5}$. The z diffusion coefficient, only ten times greater than those of x and y, is the smallest value for which time-independent waveforms have thus far been found. The unique property of this solution is that it seems to repeat itself until it fills whatever size container it is in. With the other solutions in Table 3.1, a single pulse could be stabilized no matter how long the container. Here, repetitive patterns appear. Figure 4.1 shows the evolution of such a repetitive solution from an initial condition of a single pulse.

Even more fascinating is the connection between this repetitive large-amplitude solution and the small-amplitude solutions. To see this connection, put aside the quasi-stability of the small solutions and instead focus on their shape and attractive power. For f=0.6 and $D_z=10x10^{-5}$, SMALL reveals that there is a small first-order time-independent solution. The two choices for the critical length for such a solution are 0.33351581 and 2.1309000 mm. As noted in Chapter II, any small perturbation of steady-state values in a container of the critical length evolves into a curve for each variable approaching a half wavelength of a cosine curve. Perturbations in a container of

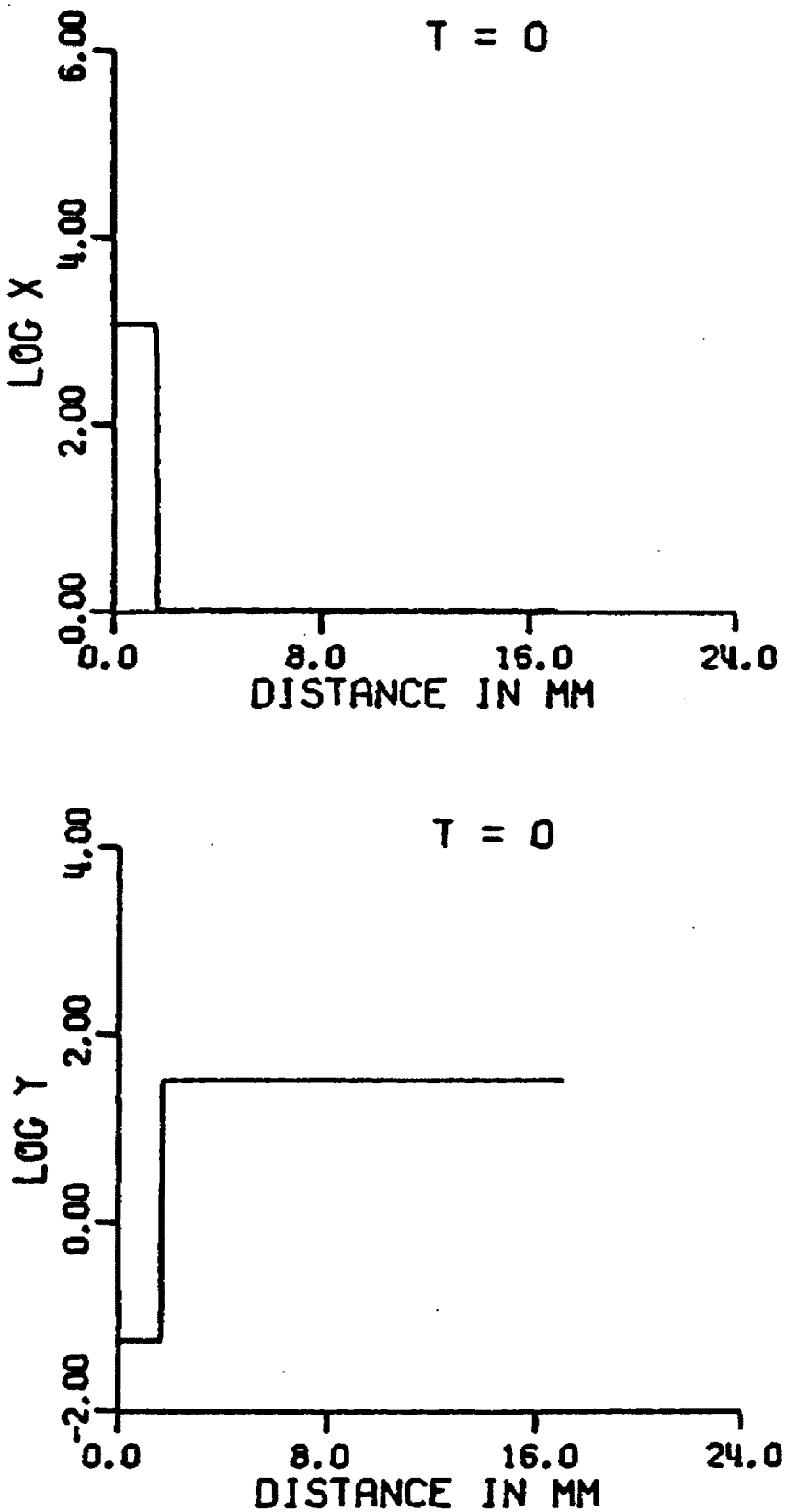
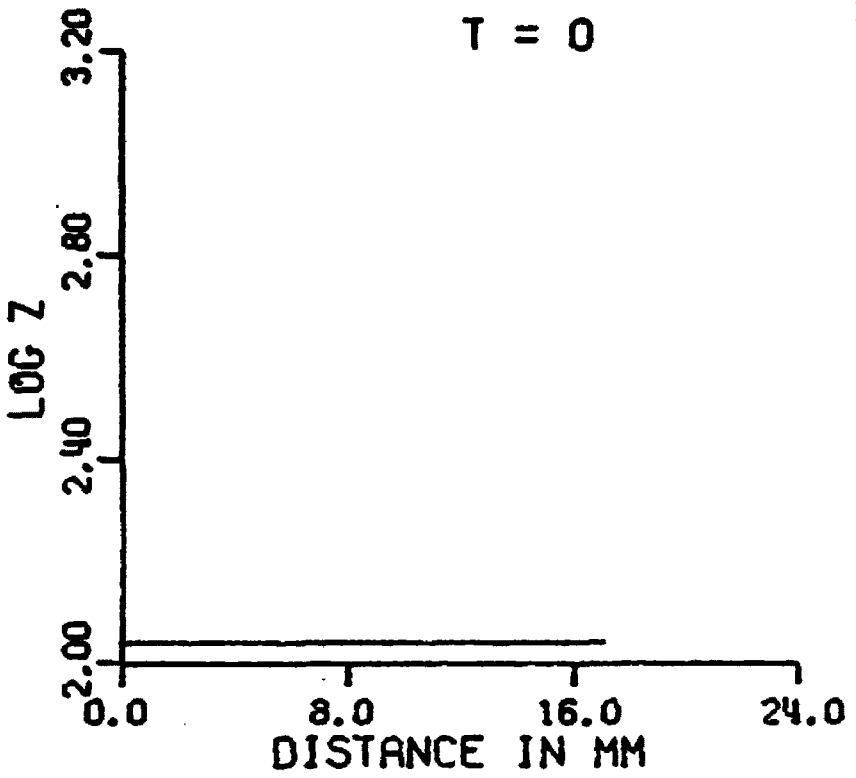


Figure 4.1: Evolution of a single pulse into a stable solution with multiple pulses. "Low" parameters.
f=0.6 $D_z=10^{-4}cm^2sec^{-1}$ L=17



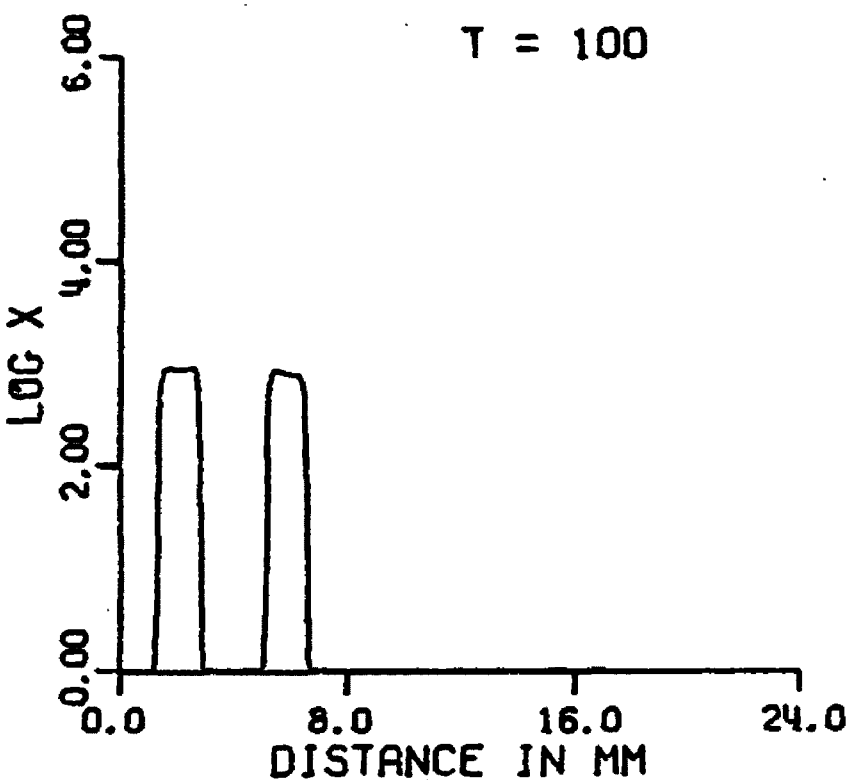
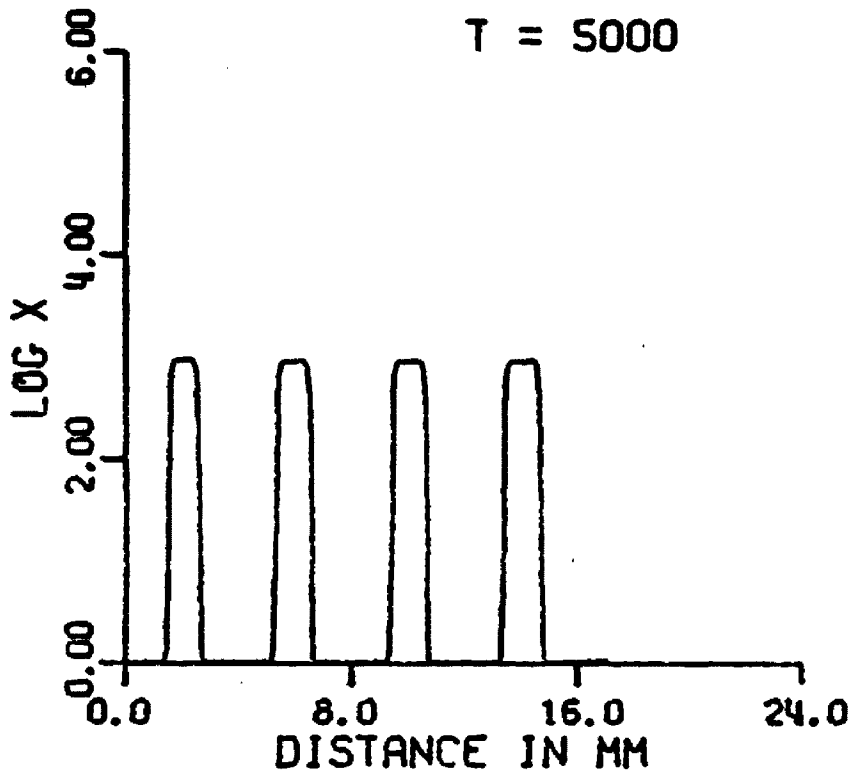


Figure 4.1 (continued)



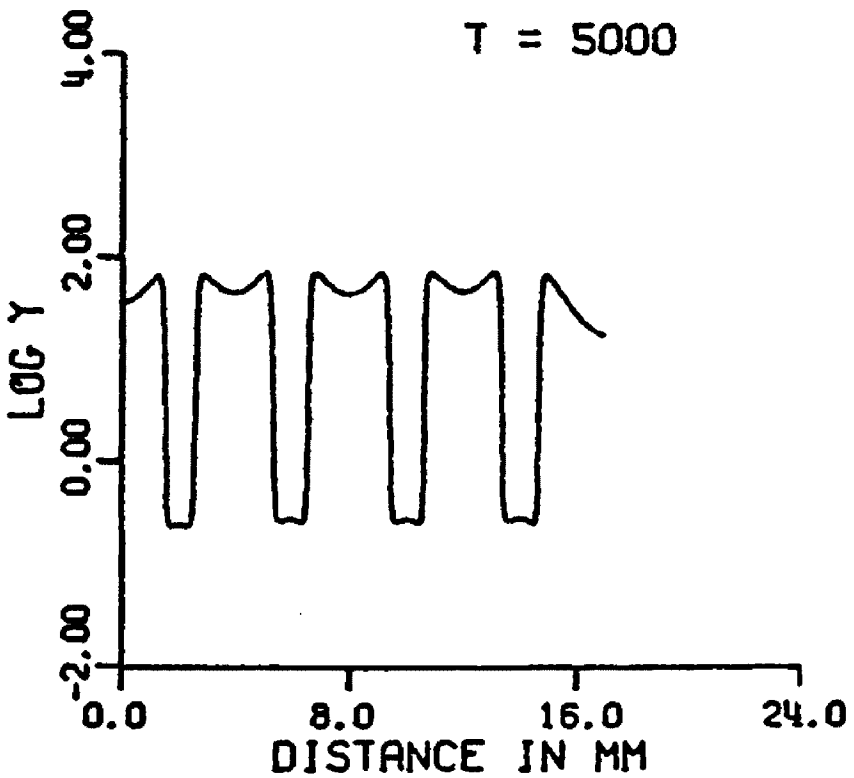
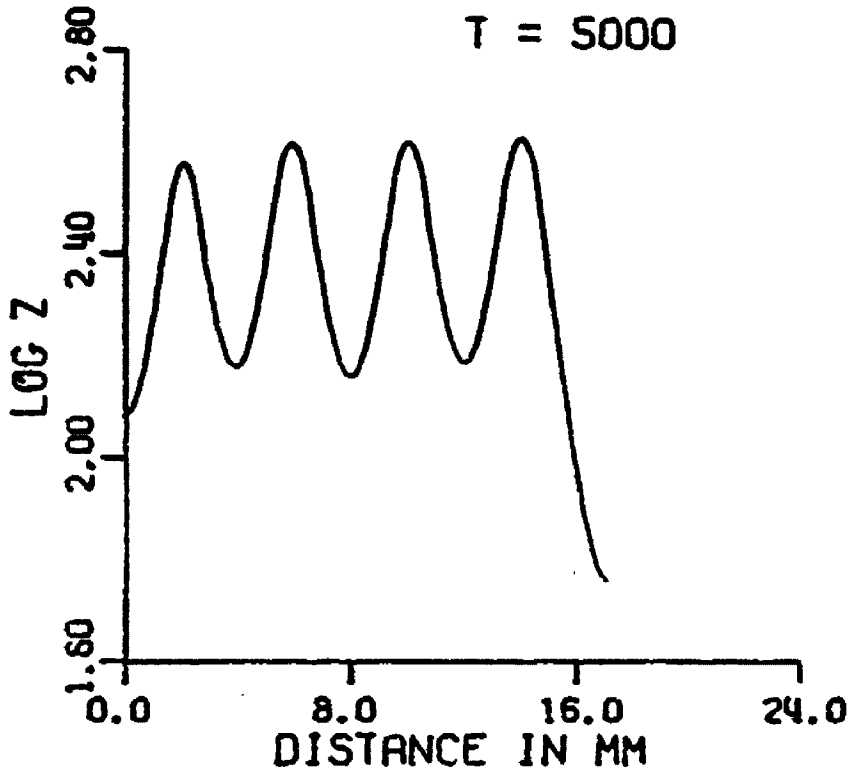


Figure 4.1 (continued)



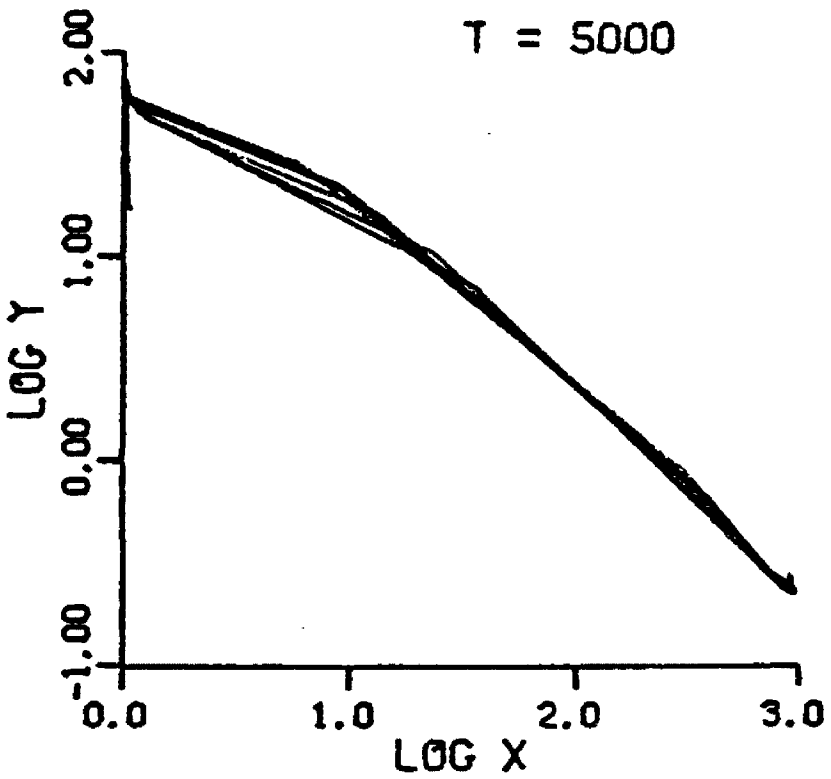


Figure 4.1 (continued)

twice the critical length seem to evolve to a whole (two half) cosine cycle. Of course these curves are not stable. A container eight times the critical length produces four full cosine waves if the proper initial perturbations are used. Moreover, any small perturbation leads to roughly four cosine cycles, or at least somewhere between three and five cycles.

It happens that for f=0.6 the critical length of 2.1309 mm for the small-amplitude solution is approximately the length of half a cycle of the large-scale stationary solution marked with ** in Table 3.1. Selecting a container length of 8x2.1309 mm and perturbing the concentrations in it in the appropriate manner described in Chapter II leads to a small-amplitude solution of four full cycles. This solution in turn grows until its peaks go very large. The large peaks are close enough to the stable large-scale solution that the curve stabilizes. Figure 4.2 shows this process. The final stable curves are slightly different than the solution pictured in Fig. 4.1, but basically the same. Ideal initial conditions were used to make this process happen, but since any initial perturbation evolves to a small waveform with roughly the proper number of peaks, it seems that any small perturbation in a container of the proper length has a good chance of evolving into a large-amplitude stable pattern. The special character of the f=0.6 and $D_z=10\times10^{-5}$ parameters, plus the similarity in the critical length of small-amplitude waveforms and the half wavelength of the large, stable solutions, create a situation where large-scale stable patterns are almost certain to result from random perturbations of a flat state.

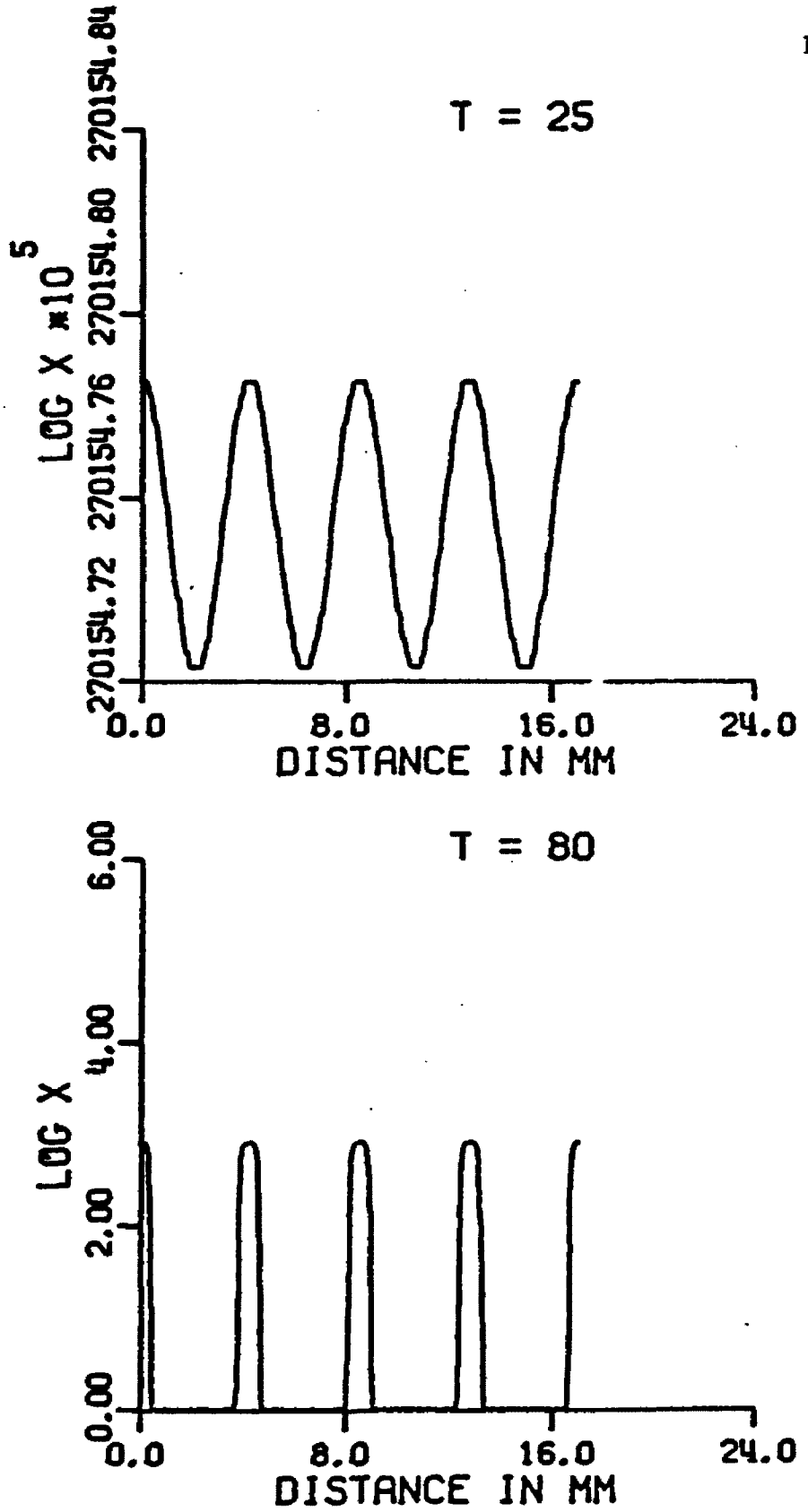
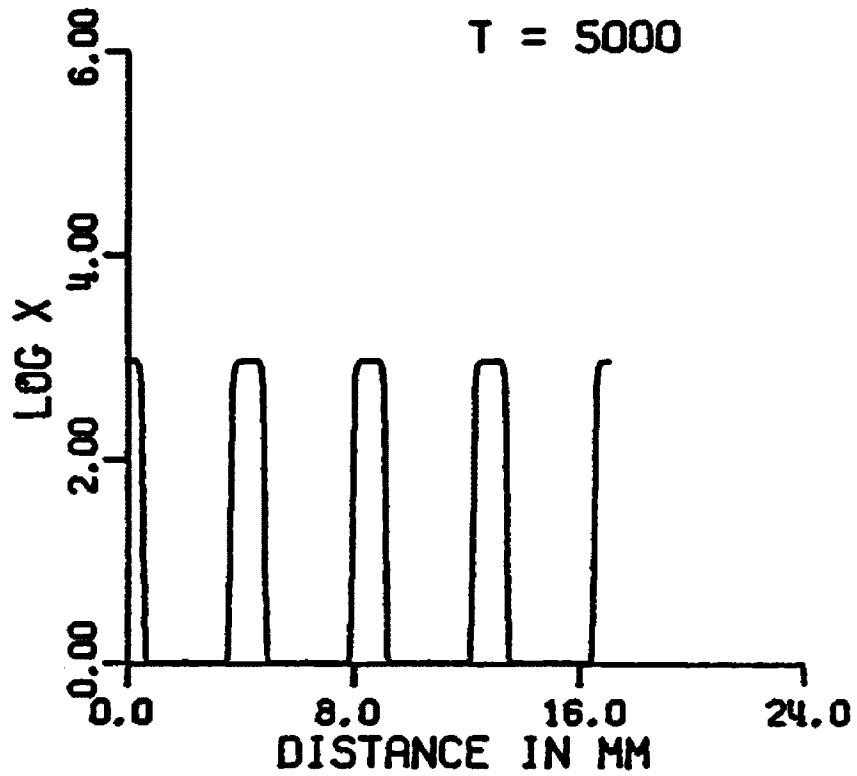


Figure 4.2: Evolution of small-amplitude multiple pulses into a large-amplitude stable solution. "Low" parameters. f=0.6 $D_z=10^{-4}$ L=17.0472



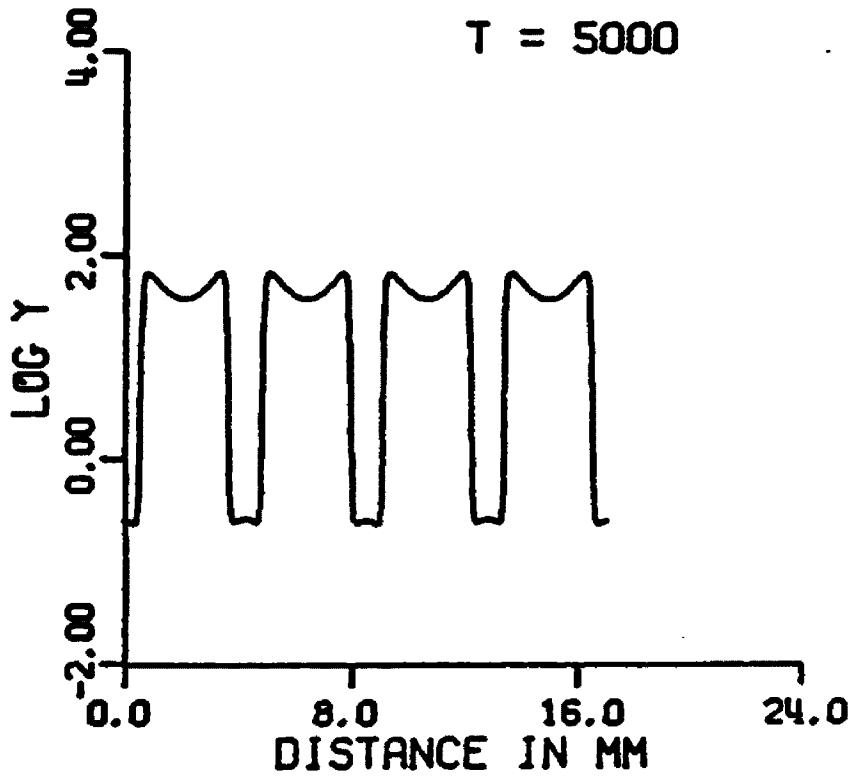
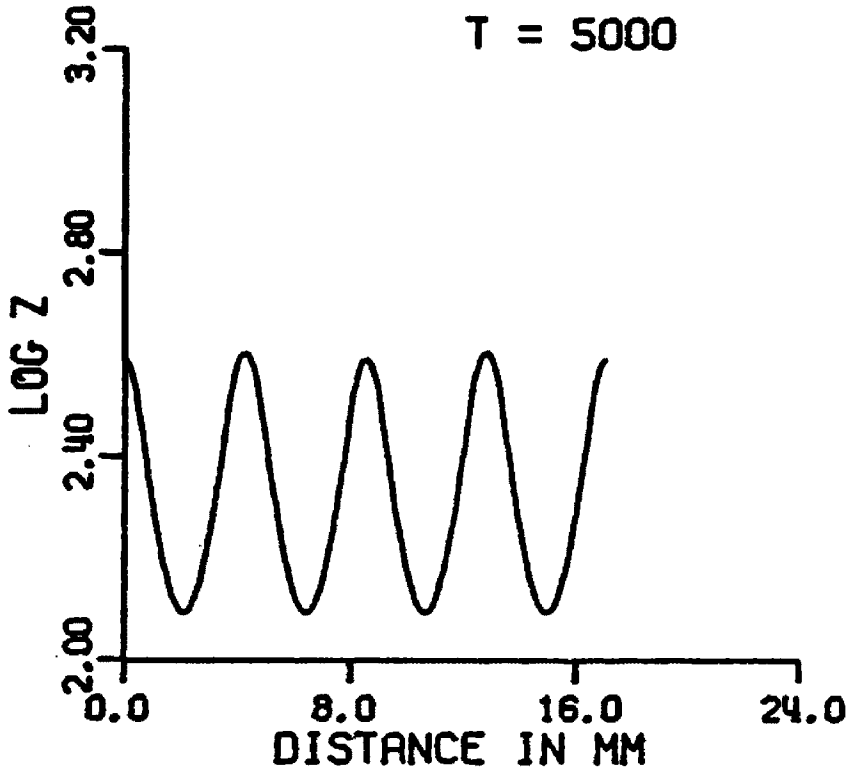


Figure 4.2 (continued)



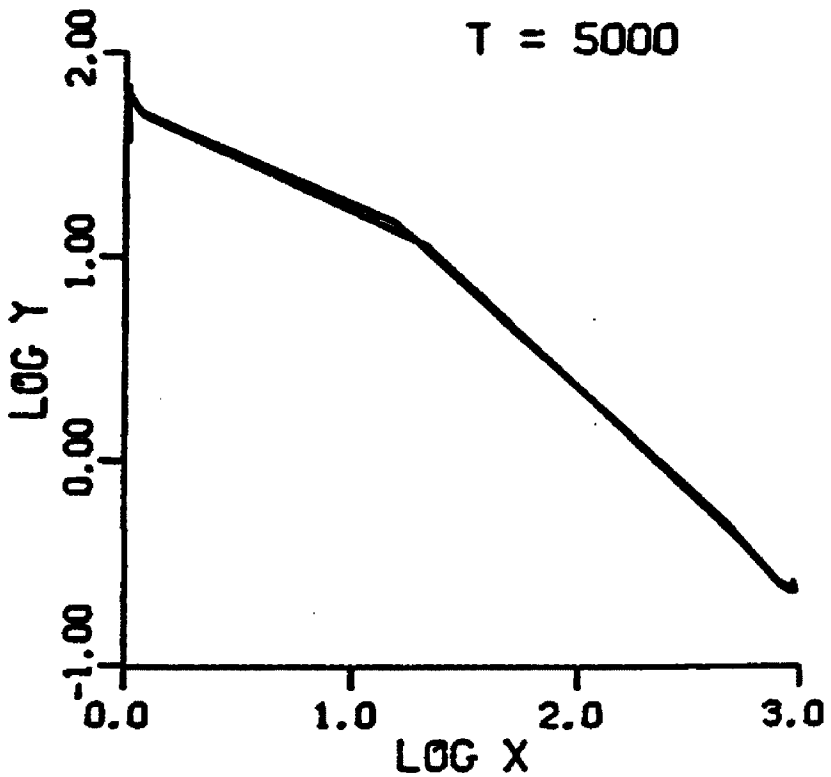


Figure 4.2 (continued)

B. SPECULATIONS

There are two areas in which speculation arising from the findings of this work is merited. The first is suggested by Fig. 4.3. Murray[15] has considered that chemical morphogens might have different diffusion coefficients than other significant chemicals in a reaction. It is known that the diffusion coefficients of macromolecules are in the realm of 100 to 1,000 times less than those of much smaller inorganic molecules. For f=1.0 and diffusion coefficients reduced to $D_x=D_y=10^{-7}$ and $D_z=10^{-5}$, Fig. 4.3 shows stable waveforms with a great increase in x concentration and a decrease in y concentration at the edges of a container. If x and y are assumed to be macromolecules in some chemical system with properties similar to the Oregonator, the chemical distribution of Fig. 4.3 is suggestive of a situation conducive to the formation of cell walls.

Second, it is interesting to relate the special large-scale waveforms (Fig. 4.2) that evolved from small perturbations to experimental work by Showalter[16] and Orbán[17]. Showalter worked with a ferroin-bromate system in a petri dish and found patterns which appeared suddenly and seemed to remain completely stable for a period of about five minutes. He attributed these patterns to a convective instability. Similar structures were observed in a related chemical system by Orbán, who ascribed them to the same cause as Showalter. Orbán supported this conclusion by describing how the placement of a glass plate on top of the petri dish stopped the patterns from appearing. The work presented

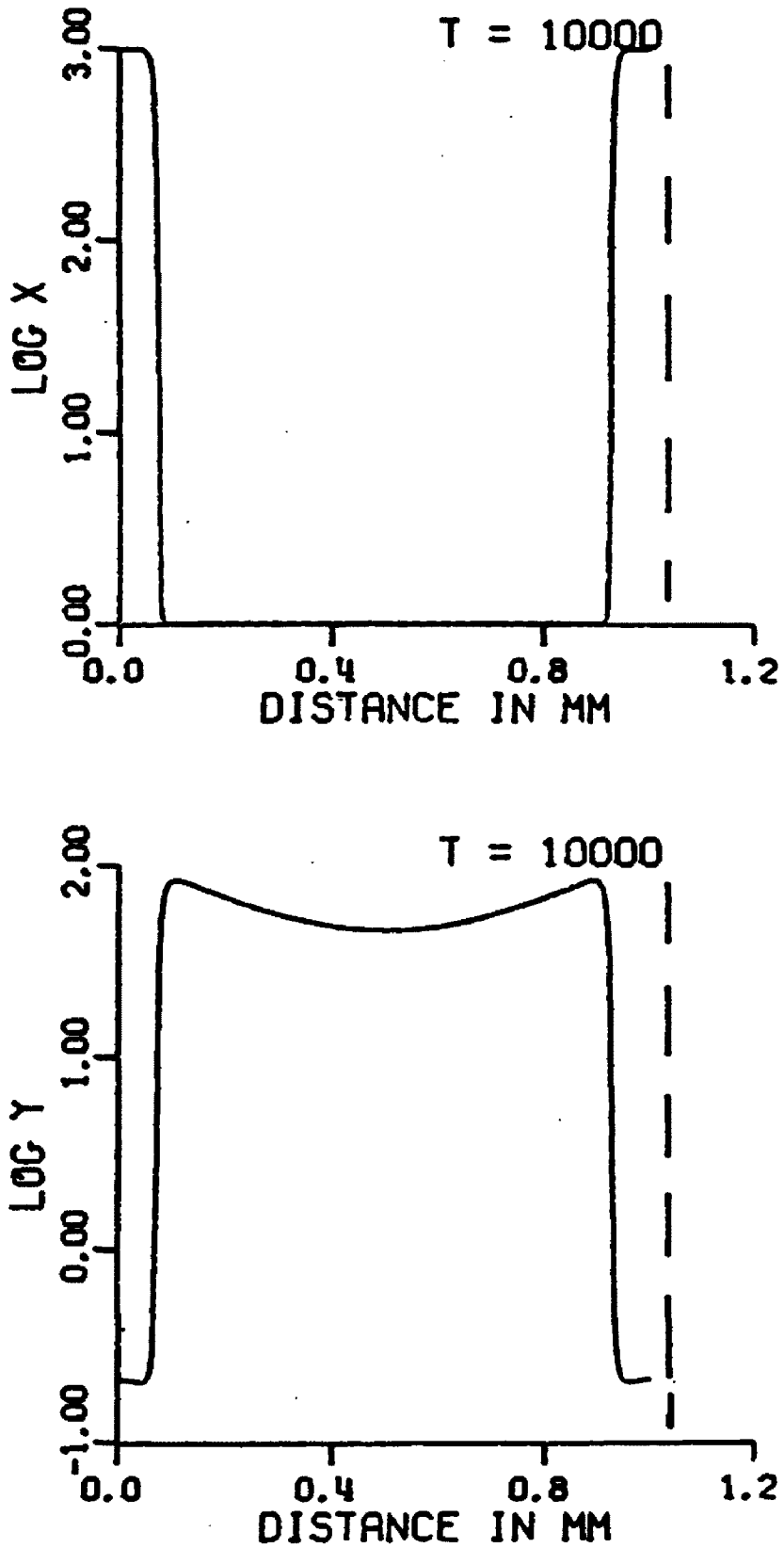


Figure 4.3: Stable solution suggestive of a cell. "Low" parameters. f=1 $D_x=D_y=10^{-7}cm^2sec^{-1}$ $D_z=10^{-5}cm^2sec^{-1}$ L=1.0

here reopens the possibility that such patterns may be caused primarily by the interaction of chemical kinetics and diffusion. Although the value of D_z is still ten times larger than physically likely, it is noteworthy that the width of the numerical pulses in Fig. 4.2 are of the same order of magnitude as those seen by Showalter and Orbán. The disappearance of the patterns when the dish is covered could be explained by a change in the parameter f due to a change in available oxygen in the space between the liquid and the glass plate[1] .

C. SUMMARY

Large-amplitude time-independent solutions to the Oregonator reaction-diffusion equations are found for the "Low" parameter set over a range of f from 0.1 to 2.5 and a range of D_z of 10^{-5} to 0.1 cm^2sec^{-1}. Similar stable solutions are also found for several other parameter sets including the Field-Noyes, or "High", values, and a set with diffusion coefficients for x and y reduced to a level appropriate for macromolecules. In x-y phase space with distance, not time, as the independent variable, the solutions stretch between the two extreme intersection points of the $x''=0$ and $y''=0$ curves (named trade lines), following these intersection points as the latter curve shifts with changes in z. Three necessary criteria are proposed for the existence of stable solutions. First, the x and y trade lines must intersect at three points for every value of z in the solution. Second, all z values must lie within the range of x values. Finally, appropriate minimum lengths for an initial pulse and for a container must be selected. As

was suggested by the phase-space diagram of Fig. 3.3, solutions are found to begin and end in the regions outlined by the intersection of the trade lines.

Small-amplitude solutions to the first and second-order perturbed reaction-diffusion equations are found, by analytical methods, to exist for critical lengths of the container. Critical lengths, L, depend on rate constants and diffusion coefficients. First-order solutions for x, y and z are each proportional to the same cosine term, with the ratio of their amplitudes fixed by an eigenvector. Small-amplitude perturbations of this form are quasi-static, changing much more slowly than perturbations in containers of arbitrary length, and more slowly than perturbations of arbitrary shape. Second-order terms do not add significantly to the lifetime of small-amplitude waveforms.

Though first-order solutions are of the form $\cos(n\pi/L)$ for any integer n, small perturbations seem to evolve toward the cosine shape with n=1. With f=0.6 and $D_Z=10\times10^{-5}$, a small perturbation is shaped sufficiently by this tendency while still at a small amplitude to grow to a stable large-amplitude solution. The existence of stable waveforms may have significance in explanations of morphogenesis, the formation of cell walls, and certain results seen in the BZ reaction.

D. WORK TO BE DONE

There are three clear areas for further research. The most pressing is to examine in more depth the ** waveform in Table 3.1. Is that waveform qualitatively different than the other stable solutions? Looking at the time-independent differential equation for z, one would think that as the value of D_z decreased, the range z takes in a stable solution would increase. Of course, if the range of z increases too much, one of the three necessary criteria for stability will be broken and stable solutions will not exist. But, as the value of D_z is decreased from 50×10^{-5} to 10×10^{-5}, the range of z does not increase as expected—rather, it decreases! Is it possible to predict analytically for a given set of parameters what the range of z will be? Further attempts to find stable solutions with D_z reduced below 10×10^{-5} should be made. Can D_z be made as small as D_x and D_y? Can it be made smaller, and will the resulting increase in the motion of the y trade line require a different type of analysis? Values of f between 0.2 and 0.9 should certainly be explored more thoroughly in an attempt to decrease D_z.

A second question to be clarified is based on the evolution from small-amplitude quasi-static waveforms to large stable forms. The small-amplitude calculations done for this work focused on the longevity of waveforms rather than on the shape which they tend to assume. Is a container of critical length necessary for random small perturbations of a flat concentration curve to evolve to a unique shape? Why, when the

container length was eight times a critical length, does an extra extrema appear or disappear? What happens when longer container lengths are used? For what parameters will small perturbations evolve into stable large waveforms?

Finally, a third direction for research is apparent. Many models for the BZ reaction involve two or three spatial dimensions. It would be interesting to know what stable waveforms exist for those higher dimensional cases.

REFERENCES

1. Field, R. J. in "Oscillations and Traveling Waves in Chemical Systems"; Field and Burger, Ed.; Wiley: New York, 1984; Chapter 2.

2. Field, R. J.; Körös, E.; Noyes, R. M. J. Amer. Chem. Soc. 1972, 94, 8649.

3. Field, R. J.; Noyes, R. M. J. Chem. Phys. 1974, 60, 1877.

4. Reusser, E. J.; Field, R. J.; J. Amer. Chem. Soc. 1979, 101, 1063.

5. Zaikin, A. N.; Zhabotinskii, A. M. Nature(London). 1970, 225, 535.

6. Field, R. J.; Noyes, R. M. J. Amer. Chem. Soc. 1974, 96, 2001.

7. Turing, A. M.. Phil. Trans. R. Soc. Lond. 1952, B 273, 37.

8. Nicolis, G.; Prigogine, I. "Self-Organization in Nonequilibrium Systems"; Wiley: New York, 1977; Chapter 7.

9. Tyson, J. J. in "Oscillations and Traveling Waves in Chemical Systems"; Field and Burger, Ed.; Wiley: New York, 1984; Chapter 3.

10. Curtiss, C. F.; Hirschfelder, J. O. Proc. Natl. Acad. Sci.(USA) 1952, 38, 235.

11. Hindmarsh, Lawrence Livermore Laboratory, Report UCID-30059, Rev. 1, "Solution of Ordinary Differential Equations Having Banded Jacobian." Program available as Acc No. 661 from Argonne Code Center; Argonne, Ill. 60439.

12. Gear, C. W. "Numerical Initial Value Problems in Ordinary Differential Equations"; Prentice-Hall: Englewood Cliffs, N.J., 1971; p. 209 ff.

13. Murray, J. D. J. Chem. Phys. 1974, 61, 3610.

14. Ermentrout, G. B.; Hastings, S. P.; Troy, W. C. "Large Amplitude Stationary Waves in an Excitable Lateral-Inhibitory Medium." Submitted for publication. Contact Dept. of Math., U. of Pittsburgh, Pittsburgh, Pa. 15261.

15. Murray, J. D. J. Theor. Biol. 1982, 98, 143.

16. Showalter, K. J. Chem. Phys. 1980, 73, 3735.

17. Orbán, Miklós. J. Amer. Chem. Soc. 1980, 102, 4311.

APPENDIX 1

PARAMETERS AND CONSTANTS

The following values are the "Low" parameters .

Parameter	Value	Units
$A \equiv [BrO^-]$	0.06	M
$B \equiv [BrO^-]$	0.06	M
D_x	$1x10^{-5}$	$cm^2 s^{-1}$
D_y	$1x10^{-5}$	$cm^2 s^{-1}$
D_z	$1x10^{-5}$	$cm^2 s^{-1}$
$[H^+]$	0.8	M
k_1	$2x[H^+]^2$	$M^{-1}s^{-1}$
k_2	$1x10^6x[H^+]$	$M^{-1}s^{-1}$
k_3	$10x[H^+]$	$M^{-1}s^{-1}$
k_4	2000	$M^{-1}s^{-1}$
k_5	0.01	$M^{-1}s^{-1}$

The following constants are then computed.

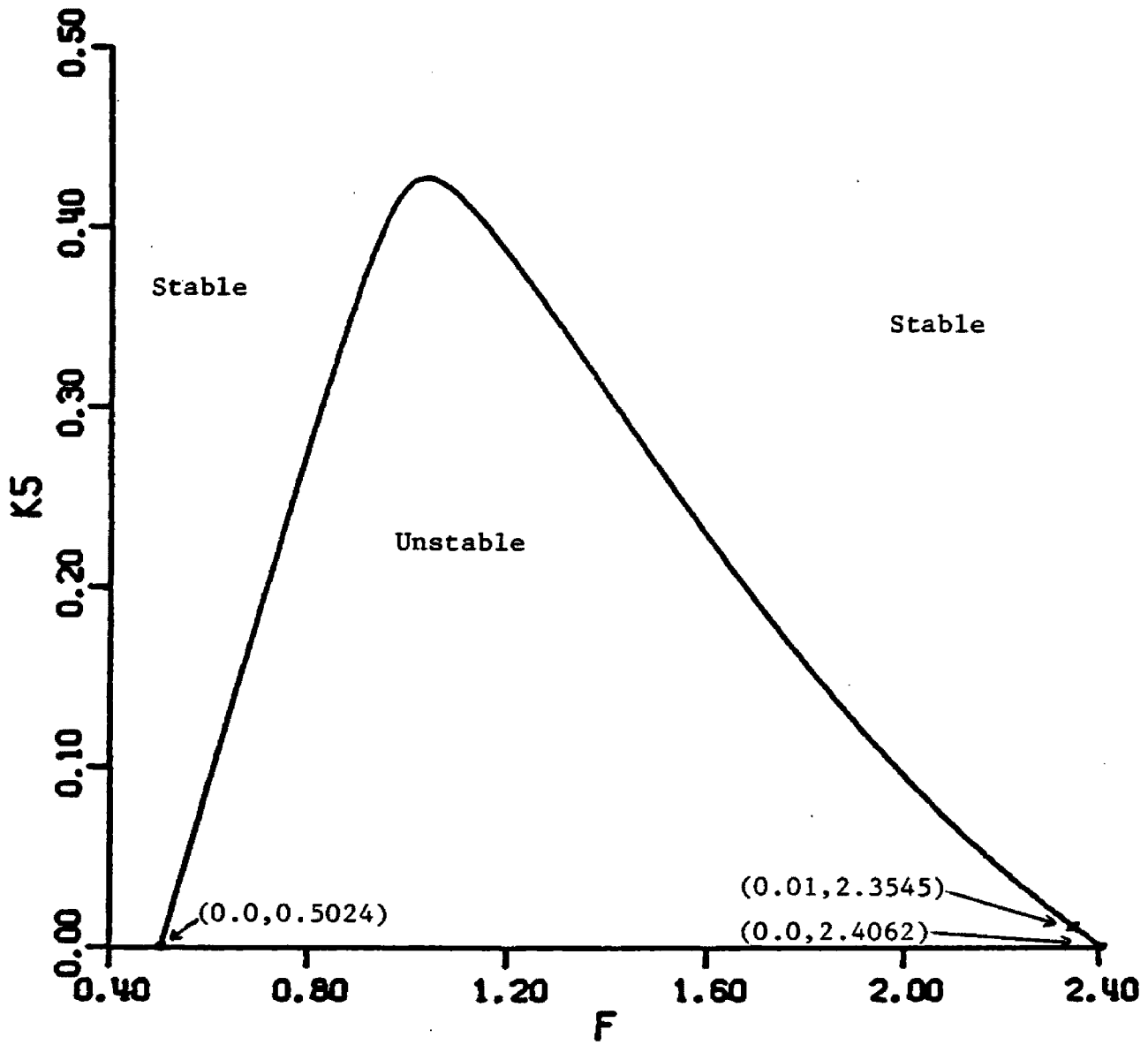
$s = 2.5$
$q = 0.0008$
$w = 0.052083$

The following are GEARB parameters.

$HO = 1x10^{-7}$
$MF = 21$
$EPS = 1x10^{-5}$

APPENDIX 2

STABILITY DIAGRAM



Calculations for the diagram above were made by the program REG.FOR (App. 3) using Eqs. 20 and 22 from a 1974 article by Murray (see Reference 13). The diagram depicts for what values of f and k_5 the steady-state solution of the basic Oregonator is stable and unstable. There are no diffusion terms, and "Low" parameters are used.

APPENDIX 3

COMPUTER PROGRAMS

THIS APPENDIX CONTAINS THE FOLLOWING PROGRAMS:

- PARSET.FOR
- NSET.FOR
- GOING.FOR
- PDB.FOR
- DIFFUN.FOR
- PATT.FOR
- INS.COS
- INS.PUL
- PICTUR.FOR
- PIC.FOR
- RUNTIM.MAC

- WIND.FOR
- REG.FOR
- SMALL.FOR
- SMALLK.FOR
- SSSOL.FOR

THE FIRST GROUP IS USED WITH TEST4, A GENERAL PACKAGE CONTAINING DRIVEB AND GEARB. PROGRAMS IN THE SECOND GROUP FUNCTION INDEPENDENTLY.

PARSET CREATES OR UPDATES FOR21.DAT, A FILE WITH EQUATION AND GEAR PARAMETERS.

NSET CREATES THE "GOING.FOR" PROGRAM WITH DIMENSIONS SET AS PER USER INPUT.

-----> GOING.FOR IN TURN DRIVES PATT. IT MUST BE INCLUDED AS THE MAIN PROGRAM WHEN EXECUTING PATT.

PDB SUBROUTINE USED BY GEAR. PDB CREATES AND EVALUATES A BANDED JACOBIAN OF THE DIFFERENTIAL EQUATIONS WHICH IS USED BY GEAR.

DIFFUN SUBROUTINE USED BY GEAR. DIFFUN EVALUATES THE SET OF 3*NOSP DIFFERENTIAL EQUATIONS.

PATT SUBROUTINE THAT RECEIVES INFO FROM USER ABOUT ADDITIONAL INPUT PARAMETERS AND ABOUT TYPE OF OUTPUT DESIRED. PATT THEN DOES AS REQUESTED. OPTIONS ARE:

1. INPUT FROM TERMINAL OR DISK FILE.
2. OUTPUTS ONLY WHEN RHO(8) CHANGES FROM RISING TO FALLING OR VICE VERSA

3. PLOTS
4. OUTPUTS TAU AND RHO(8)
5. OUTPUTS Y0 TO FOR23.DAT IN A FORM THAT CAN BE PLOTTED LATER.

INS SUBROUTINE USED BY PATT. INS, AFTER GETTING USER INPUT, RETURNS Y0 INITIALIZED. INS.COS USED FOR FIRST-ORDER SOLUTIONS. INS.PUL USEFUL FOR LARGE-AMPLITUDE SOLUTIONS.

PICTUR PICTUR PLOTS FROM DISK, GIVEN APPROPRIATE DATA FORMAT THERE.

PIC SUBROUTINE OPTIONALLY CALLED BY PATT. PIC REQUESTS USER INFO AND THEN PLOTS. USES XONOUT AND PTCNEW. CAN ALSO PRINT VALUES

SAMPLE CALLS

```
EX GOING,PATT,INS.TEST4,DIFFUN,PDB,PIC,PTCNEW,XONOUT

EX GOING,PATT,INS.TEST4,DIFFUN,PDB

EX PARSET

EX NSET
```

WIND ROUTINE TO GIVE Z VALUE AT WHICH TRADE LINES JUST TOUCH, TO GIVE COORDS OF INTERSECTION OF TRADE LINES, AND TO GRAPH TRADE LINES.

REG ROUTINE TO GRAPH REGIONS IN F-K(5) SPACE WHERE OREGONATOR WITHOUT DIFFUSION HAS STABLE AND UNSTABLE STEADY-STATES.

SMALL ROUTINE TO FIND SOLUTIONS, WHICH MEET BOUNDARY CONDITIONS, TO FIRST-ORDER REACTION-DIFFUSION OREGONATOR EQUATIONS.

SMALLK DIFFERENT VERSION OF SMALL, LETTING USER CHANGE DIF COEFFS & F.

SSSOL ROUTINE WHICH GIVES VALUES OF CONSTANTS FOR A GIVEN SET OF PARAMETERS HELD IN FOR21.DAT.

```
C PARSET.FOR

C THIS PROGRAM CREATES OR UPDATES FILE FOR21.DAT, WHICH CONTAINS
C THE PARAMETERS NAMED IN THE NAMELIST STATEMENT BELOW.
C H0, EPS AND MF ARE GEAR INITIALIZERS.  THE REST ARE THE
C PARAMETERS IN THE DIFFERENTIAL EQUATIONS.


      NAMELIST/PAR/ A,B,HPL,VK1,VK2,VK3,VK4,VK5,F,H0,EPS,MF,
     1DIFX,DIFY,DIFZ
100   READ  (21,560,ERR=200) A,B,HPL,VK1,VK2,VK3,VK4,VK5,F,H0,EPS,MF,
     1DIFX,DIFY,DIFZ
560   FORMAT (11E,I,3E)
C IF FOR21 HASN'T BEEN CREATED YET, THE ERR
C RETURN SENDS US TO OPEN THE FILE
110   TYPE 520
520   FORMAT (1H 'HI.  YOU WANT TO CHANGE PARAMETERS?'/1H
     1'PLEASE ENTER " $PAR " AND ANY PARAMETERS IN THE FORM
     2"A=2.0,MF=12" ETC., FOLLOWED BY $'/1H
     3'        DON''T FORGET THE SPACE BEFORE $PAR!')
120   READ (5,PAR,ERR=600)       !READ IN CHANGES
C THIS NAMELIST RFAD NEEDS (SP)$PAR FOLLOWED BY PARAMS AS DESCRIBED
C IN STATEMENT 520, FOLLOWED BY $.  PARAMS CAN BE IN
C ANY ORDER, ANY ONES YOU WANT TO CHANGE.
      WRITE (5,PAR)              !TYPE ALL PARAMS
      TYPE 500                   !ASK IF WE'RE DONE
500   FORMAT (1H 'DO YOU WANT TO CHANGE MORE PARAMS?'/1H
     1'IF SO TYPE 1 AND CR; THEN ENTER " $PAR " ETC.',
     2'  IF NOT, JUST HIT CR')
      READ (5,540) M
540   FORMAT (I)
      IF (M.NE.0) GO TO 120        !IF NOT DONE, TRY AGAIN
C  *  *  *  *  *  *  *

      REWIND 21                    !WE'RE DONE.  SAVE PARAMS.
      WRITE (21,560)A,B,HPL,VK1,VK2,VK3,VK4,VK5,F,H0,EPS,MF,
     1DIFX,DIFY,DIFZ
      STOP
C  *  *  *  *  *  *  *  *  *
200   OPEN(UNIT=21,ACCESS= 'SEQOUT',FILE='FOR21.DAT') !OPEN FILE HERE
      GO TO 110
C  *  *  *  *  *  *  *
600   TYPE 580
580   FORMAT(1X'ERROR IN ENTERING DATA. TRY AGAIN')
      GO TO 120
      END
```

```
C NSET.FOR

C  NSET CREATES "GOING," THE MAIN PROGRAM CONSISTING OF STATEMENTS TO
C  SET DIMENSIONS, AND A CALL TO PATT, THE WORKHORSE SUBROUTINE


      WRITE (5,510)
510   FORMAT(' NO. OF SPACE PTS?')
      READ (5,500) NOSP           !INPUT IS NOSP
500   FORMAT (I)
      N=3*NOSP
      NPLT=NOSP+2    !ARRAYS FOR PLOT HAVE 2 EXTRA WORDS
      M=10*N
      OPEN(UNIT=24,ACCESS='SEQOUT',FILE='GOING.FOR')

C  Y0 AND Y ARE FOR GEAR; YWAS IS FOR PATT; ZYPIC AND ZDIS ARE REAL
C  ARRAYS FOR PIC

      WRITE (24,520)N,NPLT,M,NOSP
520   FORMAT(7X,'IMPLICIT DOUBLE PRECISION(A-H,O-Y)'/
      17X,'PARAMETER N='I5,',   PLT='I5,',   M='I5,',   NPPP='I5/
      17X,'DIMENSION Y0(N),Y(N,6),YWAS(N),ZYPIC(PLT),ZDIS(PLT)'/
      27X,'COMMON /GEAR2/YMAX(N)'/
      37X,'COMMON /GEAR3/ERROR(N)'/
      47X,'COMMON /GEAR4/SAVE1(N)'/
      57X,'COMMON /GEAR5/SAVE2(N)'/
      67X,'COMMON /GEAR6/ZPW(M)'/
      77X,'COMMON /GEAR7/IPIV(N)'/
      87X,'NOSP=NPPP'/
      97X,'CALL PATT(NOSP,Y0,Y,YWAS,ZYPIC,ZDIS)'/
      97X,'STOP'/7X,'END')
      END
```

C GOING.FOR

```
      IMPLICIT DOUBLE PRECISION(A-H,O-Y)
      PARAMETER N=  750,   PLT=  252,   M= 7500,   NPPP=  250
      DIMENSION YO(N),Y(N,6),YWAS(N),ZYPIC(PLT),ZDIS(PLT)
      COMMON /GEAR2/YMAX(N)
      COMMON /GEAR3/ERROR(N)
      COMMON /GEAR4/SAVE1(N)
      COMMON /GEAR5/SAVE2(N)
      COMMON /GEAR6/ZPW(M)
      COMMON /GEAR7/IPIV(N)
      NOSP=NPPP
      CALL PATT(NOSP,YO,Y,YWAS,ZYPIC,ZDIS)
      STOP
      END
```

```
C PDB.FOR

C       PDB IS CALLED BY PSETB IN GEAR.
C       TO PARAPHRASE THE DESCRIPTION IN DRIVEB OF GEAR:
C  PDB COMPUTES THE N BY N BANDED JACOBIAN MATRIX OF PARTIAL
C  DERIVATIVES AND STORES IT IN ZPD AS AN N BY 7 ARRAY.
C  ZPD(I,J-I+4) IS TO BE SET TO THE PARTIAL DERIVATIVE OF YDOT(I)
C  WITH RESPECT TO Y(J).


C       NOW IN MORE DETAIL
C             LETTING A STAND FOR ALPHA
C                     E STAND FOR ETA
C                     R STAND FOR RHO
C               AND SREC =RECIPROCAL OF S
C  OUR GENERAL EQUATIONS ARE:

C ADOT(I)=(D'/DX**2)*[A(I-1)-2*A(I)+A(I+1)]+S*[E(I)+A(I)*(-E(I)+1-Q*A(I]

C EDOT(I)=(D'/DX**2)*[E(I-1)-2*E(I)+E(I+1)] +SREC*[F*R(I)-E(I)*(1+A(I))]

C RDOT(I)=(D'/DX**2)*[R(I-1)-2*R(I)+R(I+1)]  +W*A(I)-W*R(I).

C  SINCE Y GOES TO NOSP, WE'VE DESCRIBED 3*NOSP SIMULTANEOUS EQUATIONS.
C  NOW, LOOK AT THE RIGHT SIDE OF EACH EQ AND TAKE IT'S P.D. WITH
C  RESPECT TO EVERY A(I),E(I) & R(I).  MOST OF THESE PD
C  ARE CLEARLY ZERO.  FOR A GIVEN I, ONLY THOSE PD INVOLVING
C  I-1, I, OR I+1 HAVE A CHANCE TO BE NON-0.  WE STORE THESE REMAINING
C  PD IN ZPD IN THE ODD WAY DESCRIBED ABOVE.
C     DO NOTE THAT THE FOLLOWING ARE NOT SPECIFIED IN OUR
C  INSTRUCTIONS:  ZPD(1,1), (1,2), (1,3), (2,1), (2,2), (3,1),
C  (N-2,7), (N-1,6), (N-1,7), (N,5), (N,6), & (N,7).
C  THEY ARE NOT USED BY PDB SO CAN BE SET TO ANYTHING CONVENIENT.
C  THE FIRST FEW MEANINGFUL ENTRIES OF ZPD THEN ARE:
C  (1,4)= PD ADOT(1) WITH RESPECT TO A(1)
C  (1,5)= PD ADOT1 W-R-T E1
C  (1,6)= PD ADOT1 W-R-T R1
C  (1,7)= PD ADOT1 W-R-T A2
C  (2,3)= PD EDOT1       A1
C          ETC.
C  (4,1)= PD ADOT2       A1
C  (4,2)= PD ADOT2       E1
C  (4,3)= PD ADOT2       R1
C  (4,4)= PD ADOT2       A2
C  (4,5)= PD ADOT2       E2
C  (4,6)= PD ADOT2       R2
C  (4,7)= PD ADOT2       A3
C  (5,1)= PD EDOT2       E1
C         ETC
```

```
C  (5,7)= PD EDOT2       E3
C         ETC
C  PD FOR ALL POINTS EXCEPT THE BOUNDARY POINTS ARE HANDLED DOWN TO
C  STATEMENT 10.  THE REST OF THE PROGRAM MODIFIES THE BOUNDARIES.

C       WE ASSUME NO DIFFUSION AT BOUNDARIES.  THIS IS LIKE SAYING THE
C  POINT PAST THE BOUNDARY ALWAYS HAS THE SAME CONCENTRATIONS AS THE
C  BOUNDARY.
C  AT 0 BOUND [A(0)-2*A(1)+A(2)] BECOMES [-A(1)+A(2)] SINCE A(0)=A(1).
C  AT N BOUND, [A(N-1)-2*A(N)+A(N+1)] BECOMES [A(N-1)-A(N)] SINCE
C  A(N+1)=A(N).   SIMILARLY FOR E AND R.
C  THE ONLY PD'S THAT CHANGE FROM OUR GENERAL FORM ARE:
C  ADOT1, EDOT1, & RDOT1 W-R-T A1, E1, & R1 RESPECTIVELY, AND
C  ADOTN, EDOTN, & RDOTN W-R-T AN, EN & RN RESPECTIVELY.
C  THESE ARE ADJUSTED BY STATEMENTS BELOW NUMBER 10
C  ALL COMMON CONSTANTS COME FROM PATT.


      SUBROUTINE PDB (N,T,Y,ZPD,NO,ML,MU)
C             WE USE ONLY N, ZPD AND Y(1,1) TO Y(N,1).
C             SEE STIFFB & DRIVEB FOR DESCRIPTION OF OTHER PARAMS.
      IMPLICIT DOUBLE PRECISION(A-H,O-Y)
      DIMENSION Y(1), ZPD(N,1)
      COMMON /PATT1/ FF,Q,S,SREC,W,DIFXOV,DIFYOV,DIFZOV !SEE PATT.
      COMMON/GO1/DX
      DO 10 I=1,N-2,3
      J=I+1
      K=I+2
      ZPD(I,1)=DIFXOV
      ZPD(J,1)=DIFYOV
      ZPD(K,1)=DIFZOV
      ZPD(I,2)=0.
      ZPD(J,2)=0.
      ZPD(K,2)=W
      ZPD(I,3)=0.
      ZPD(J,3)=-SREC*Y(J)
      ZPD(K,3)=0.
      ZPD(I,4)=-2.*DIFXOV+S*(-Y(J)+1.-2.*Q*Y(I))
      ZPD(J,4)=-2.*DIFYOV-SREC*(1.+Y(I))
      ZPD(K,4)=-2.*DIFZOV
      ZPD(I,5)=S*(1.-Y(I))
      ZPD(J,5)=FF*SREC
      ZPD(K,5)=0.
      ZPD(I,6)=0.
      ZPD(J,6)=0.
      ZPD(K,6)=0.
      ZPD(I,7)=DIFXOV
      ZPD(J,7)=DIFYOV
      ZPD(K,7)=DIFZOV
10    CONTINUE
```

```
ZPD(1,4)=ZPD(4,4)+DIFXOV
ZPD(2,4)=ZPD(5,4)+DIFYOV
ZPD(3,4)=ZPD(6,4)+DIFZOV
ZPD(N-2,4)=ZPD(N-5,4)+DIFXOV
ZPD(N-1,4)=ZPD(N-4,4)+DIFYOV
ZPD(N,4)=ZPD(N-3,4)+DIFZOV
RETURN
END
```

DIFFUN.FOR

```
C  DIFFUN, CALLED BY STIFFB IN GEAR (NEAR #110 AND #400), COMPUTES
C  THE VALUES OF OUR 3 DIFFERENTIAL EQUATIONS AT EVERY SPACE POINT.
C  REMEMBER Y0 IS ARRANGED IN THE FORM A(1),E(1),R(1),A(2),E(2),...

C       WE ASSUME NO DIFFUSION AT BOUNDARIES.  THIS IS LIKE SAYING THE
C  POINT PAST THE BOUNDARY ALWAYS HAS THE SAME CONCENTRATIONS AS THE
C  BOUNDARY.
C  AT 0 BOUND [A(0)-2*A(1)+A(2)] BECOMES [-A(1)+A(2)] SINCE A(0)=A(1).
C  AT N BOUND, [A(N-1)-2*A(N)+A(N+1)] BECOMES [A(N-1)-A(N)] SINCE
C  A(N+1)=A(N).   SIMILARLY FOR E AND R.
      SUBROUTINE DIFFUN(N,T,Y,YDOT)
      IMPLICIT DOUBLE PRECISION (A-H,O-Y)
      COMMON /GO1/ DX
      COMMON /PATT1/ FF,Q,S,SREC,W,DIFXOV,DIFYOV,DIFZOV !SEE PATT
      DIMENSION Y(1),YDOT(1)

C  FIRST SET BOUNDARY CONDITIONS.   *  *  *  *  *  *
      YDOT(1)=DIFXOV*(Y(4)-Y(1)) +S*(Y(2)+Y(1)*(-Y(2)+1.-Q*Y(1)))
      YDOT(2)=DIFYOV*(Y(5)-Y(2)) +SREC*(-Y(2)*(1.+Y(1)) +FF*Y(3))
      YDOT(3)=DIFZOV*(Y(6)-Y(3)) +W*(Y(1)-Y(3))
      YDOT(N-2)=DIFXOV*(Y(N-5)-Y(N-2)) +S*(Y(N-1)+Y(N-2)*(-Y(N-1)+
     1  1.-Q*Y(N-2)))
      YDOT(N-1)=DIFYOV*(Y(N-4)-Y(N-1))+SREC*(-Y(N-1)*(1.+Y(N-2))
     1  +FF*Y(N))
      YDOT(N)=DIFZOV*(Y(N-3)-Y(N)) +W*(Y(N-2)-Y(N))
C                                  *  *  *  *  *  *
C  NOW COMPUTE GENERAL VALUES
      DO 10 I=4,N-5,3
      YDOT(I)=DIFXOV*(Y(I+3)-2.*Y(I)+Y(I-3))+
     1  S*(Y(I+1)+Y(I)*(-Y(I+1)+1.-Q*Y(I)))
      YDOT(I+1)=DIFYOV*(Y(I+4)-2.*Y(I+1)+Y(I-2))
     1  +SREC*(-Y(I+1)*(1.+Y(I))+FF*Y(I+2))
      YDOT(I+2)=DIFZOV*(Y(I+5)-2.*Y(I+2)+Y(I-1))
     1  +W*(Y(I)-Y(I+2))
10    CONTINUE
      RETURN
      END
```

```
C PATT.FOR

C  PATT,CALLED BY GOING, IS THE PRINCIPLE ROUTINE FOR DRIVING GEAR.

C  PATT HAS SEVERAL VARIATIONS WHICH CAN BE EASILY ENABLED BY THE
C  FOLLOWING MODIFICATIONS:
C  1. SETTING PARAMETER IN=22 WILL READ INPUT FROM FOR22.DAT RATHER
C     THAN FROM TERMINAL.  YOU MUST ALSO SET IN=22 IN ROUTINE INS
C     AND, IF YOU'RE PLOTTING, IN ROUTINE PIC.  IF YOU DON'T WANT ANY
C     MESSAGES TO TERMINAL (EXCEPT ERRORS), CHANGE "TYPE" TO
C     "CONTINUE!" IN ALL THREE ROUTINES.

C  2. THE EDIT COMMAND SC6$66$3200:* SELECTS (FOR OUTPUTTING) ONLY A
C     RETURN FROM GEAR IN WHICH RHO(8) CHANGES FROM RISING TO FALLING
C     OR VICE VERSA.  OPTION 4 BELOW THEN WILL PRINT RELATIVE MAXIMUMS
C     AND MINIMUMS OF RHO(8).  OPTIONS 3 & 5 WILL OUTPUT YO ONE
C     TIME-INCREMENT AFTER RHO(8) REACHES MAX OR MIN.  NORMALLY
C     USED WITH A TINC (TIME INCREMENT) TYPE INPUT.

C  3. SC5$55$3200:* WILL SEND OUTPUT TO THE PLOTTER.

C  4. SC7$77$3200:* WILL OUTPUT TAU AND RHO(8) TO DEVICE "OUT".
C     OUT IS SET IN PARAMETER STATEMENT [5 (TTY) OR 23 (FOR23.DAT)]

C  5. SC8$88$3200:* WILL OUTPUT SOME PARAMS AND YO TO DEVICE "OUT"
C     (WHICH SHOULD BE FOR23.DAT) IN A FORM THAT ROUTINE PICTUR
C     WILL USE AS INPUT FOR LATER PLOTTING.

C  NOTE THEN THAT STATEMENT NUMBERS BEGINNING WITH 55,66,77,OR 88
C  SHOULD NOT BE ADDED TO THIS PROGRAM, OR THE ABOVE OPTIONS WILL SCREW
C  THEM UP

C  NOSP AND ARRAYS PASSED FROM GOING
      SUBROUTINE PATT(NOSP,YO,Y,YWAS,ZYPIC,ZDIS)
      IMPLICIT DOUBLE PRECISION(A-H,O-Y)   !Z IS REAL
      PARAMETER IN=5,OUT=23
      INTEGER RUNTIM
      DIMENSION YO(1),Y(1,6),YWAS(1),ZYPIC(1),ZDIS(1)
      COMMON /PIC2/ N,TOUT,IRUN   !NOSP*3, OUTPUT TIME, RUN #
      COMMON /GO1/ DX
      COMMON /PATT1/ FF,Q,S,SREC,W,DIFXOV,DIFYOV,DIFZOV
      COMMON /INS1/ TOTL

C  *  *  *  *  *  *  *  *     INITIALIZING
      CALL ERRSET(0)  !WE HAVE ABOUT 2 MILLION FL. PT. UNDERFLOWS
551   CALL INITAL(5,254,11,1,0,0)  !(OPTION 3. INITIALIZE PLOTTER)
552   CALL RSTR(0)
553   READ (5,150)IDUMMY            ! DUMMY READ AFTER RSTR
      IRUN=0                        !RUN-NUMBER FOR PLOTTER
```

```
      N=NOSP*3                          !NUMBER OF ENTRIES IN YO
      NPLT=NOSP+2                 !TWO EXTRA WORDS IN PLOTTING ARRAYS

C  *  *  *  *  EVERY RESTART COMES HERE FOR INITIALIZING
C      (SO YOU CAN ALTER FOR21.DAT BEFORE A RESTART).
100   REWIND 21                         !(FOR21.DAT SET BY PARSET)
      READ (21,110,ERR=460) A,B,HPL,VK1,VK2,VK3,VK4,VK5,F,H0,
      1  EPS,MF,DIFX,DIFY,DIFZ          !H0, EPS, MF ARE GEAR PARAMS
110   FORMAT(11D,I,3D)
      H0SAV=H0                    !WE'LL NEED OLD H0. (H0 RESET BY GEAR)
      VK1=VK1*HPL*HPL             !SET CONSTANTS FOR DIF. EQ.'S
      VK2=VK2*HPL                !DIFFUN & PDB GET THESE THRU COMMON PATT1
      VK3=VK3*HPL
      FF=F
      Q=2.*VK1*VK4*A/(VK2*VK3*B)
      S=SQRT(VK3*B/(VK1*A))
      SREC=1./S
      CON=SQRT(VK1*VK3*A*B)
      W=VK5/CON
      DIFAL = 100.*DIFX/CON
      DIFET = 100.*DIFY/CON
      DIFRO = 100.*DIFZ/CON
      ML=3                        !WIDTHS OF BANDED JACOBIAN - SEE GEAR
      MU=3
C  *  *  *  *  *  *  *  *  *      DONE INITIALIZING


130   IRUN=IRUN+1
150   FORMAT(G)

160   TYPE 170                          ! OTHERWISE, READ INFO
170   FORMAT(1X'ENTER TAU''S: INITIAL,OUTPUT,& INC (OPT)')
C  IF TINC=0., WE'LL CALL DRIVEB WITH INPUT & OUTPUT TIMES TO & TOUT.
C  IF TINC.NE.0., TFIN IS SET TO TOUT AND SUCCESSIVE CONTINUE CALLS TO
C  DRIVEB WILL BE MADE WITH TOUT=TO+K*TINC UNTIL TFIN IS REACHED.
C  THE FINAL CALL WILL BE MADE WITH THE LARGEST TOUT.LE.TFIN.
      READ (IN,180)TO,TOUT,TINC
      IF (TOUT.LT.TO) GO TO 160
      TFIN=TOUT-1.     !SET TFIN INITIALLY FOR ONLY 1 CALL TO DRIVEB
      IF (TINC.EQ.0.) GO TO 190         !BR IF ONLY 1 CALL
      TFIN=TOUT                         ! MULTIPLE CALLS.  SET TFIN
      TOUT=TO+TINC                      ! INITIALIZE TOUT.
180   FORMAT(3D)
190   TYPE 200
200   FORMAT(1X'ENTER LENGTH (IN MM.)')    !LENGTH
      READ (IN,180)TOTL
      IF (TOTL.LE.0.) GO TO 190
      DIFXOV=DIFAL                      ! USED ONLY TEMPORARILY BY
      DIFYOV=DIFET                      ! INS.COS FOR 2ND ORDER TERM.
      DIFZOV=DIFRO
```

```
      CALL INS(NOSP,YO,ISTAND)          ! GET INITIAL YO. (ISTAND.NE.1)
290   INDEX=1                           !GEAR IS GETTING A FIRST CALL
      DX=TOTL/(NOSP-1)                  ! DX
      CON2=100./(CON*DX*DX)
C  DIFXOV IS DIFX CHANGED FROM CM**2 TO MM**2, NORMALIZED, AND DIVIDED
C  BY DX**2.  THESE NUMBERS ARE USEFUL FOR PDB & DIFFUN
      DIFXOV=DIFAL/(DX*DX)
      DIFYOV=DIFET/(DX*DX)
      DIFZOV=DIFRO/(DX*DX)

C  -  -  -  -  -  -  -  -  -  -   CONTINUE OR OMIT-LAST-RUN COMES HERE
300   DO 310 I=1,N
310   YWAS(I)=YO(I)              !SAVE YO AND TO
      TSAV=TO
      CALL DRIVEB(N,TO,HO,YO,Y,TOUT,EPS,MF,INDEX,ML,MU)    !DRIVEB
      IF (INDEX.NE.0) GO TO 440                !BR IF GEAR ERROR
      TO=TOUT                                  !OTHERWISE SET NEW TO

C  CHECK THAT CPU TIME LIMIT NOT EXCEEDED WHEN USING TINC.
C  0 MEANS NO LIMIT.
      IF (TIMLT.EQ.0.) GO TO 311
      IF (RUNTIM(WORK)/1000..LE.TIMLIM) GO TO 331
      TFIN=TOUT-1.
311   CONTINUE
C60   IF ((YWAS(24)-YO(24))*SIGN.GE.0.) GO TO 662 !OPTIONS DESCRIBED
550   CALL PIC(YO,NPLT,ZYPIC,ZDIS)               !ABOVE
      CPU=RUNTIM(WORK)/1000.
C70   WRITE (5,317) TOUT,YWAS(24)                 !
C80   WRITE (OUT,316) N,TOUT,IRUN,DX,CPU              !
C81   WRITE (OUT,315) (YO(K),K=1,N)               !
315   FORMAT(E10.4)                              !
316   FORMAT (I,D,I,2D)                          !
317   FORMAT(1X'TAU= 'F8.2,' LOG(RHO(8))= 'E10.4) !
C62   SAVSIN=SIGN                                !
C61   SIGN=YWAS(24)-YO(24)                       !
331   INDEX=0                  !ASSUME WE'VE A CONTINUE USING TINC
      TOUT=TOUT+TINC
      IF (TOUT.LE.TFIN) GO TO 300 ! BR IF THAT'S TRUE.
      TOUT=TOUT-TINC    !OTHERWISE SET TOUT TO FINAL OUTPUT VALUE
332   TYPE 340
340   FORMAT(1X'0=CONTINUE;1=RESTART; 2=OMIT LAST RUN; 3=END')
      READ (IN,150,ERR=332)INDEX                 !DECIDE WHAT'S NEXT
      GO TO (100,350,410)INDEX
      TO=TOUT                  !0  COMES HERE
      INDEX=0                  !   PREPARE A CONTINUE CALL TO GEAR
      GO TO 370                !   SKIP DOWN
350   INDEX=1                  !2  OMIT LAST RUN
C63   SIGN=SAVSIN              !   RESET HO, TO, YO
      HO=HOSAV
      TO=TSAV
```

```
      DO 360 I=1,N
360   YO(I)=YWAS(I)
370   TYPE 380,TO              !0 OR 2  GET TOUT & OPTIONALLY TINC
380   FORMAT (1X'START IS 'F12.2,'  ENTER OUTPUT TAU & INC (OPT)')
390   READ (IN,180)TOUT,TINC
      IF (TOUT.LT.TO) GO TO 370
      TFIN=TOUT-1.             !   ASSUME ONLY 1 CALL TO DRIVEB
      TIMLT=0.
      IF (TINC.EQ.0) GO TO 300!   BR IF TRUE
      WRITE (5,500)            !   OTHERWISE SET UP MULTIPLE CALLS.

C  WITH MULT. CALLS, CPU TIME LIMIT POSSIBLE. 0 MEANS NO LIMIT.
500   FORMAT (' ENTER CPU |LIMIT ',$)
      READ (IN,150) TIMLT
      TIMLIM=TIMLT+RUNTIM(WORK)/1000.
      TFIN=TOUT
      TOUT=TO+TINC
      GO TO 300                !   AND DO IT

410   STOP                     !3  DONE

C  *  *  *  *  *  *  *  *  *      GEAR ERROR
440   WRITE (5,450) INDEX
450   FORMAT(1X'GEAR ERROR. INDEX='I4)
      STOP
C  *  *  *  *  *  *  *  *  *      FOR21.DAT MISSING
460   PAUSE  'WHERE IS FOR21.DAT WITH PARAMS?  USE PARSET'
      GO TO 100       !CREATE IT USING PARSET AND THEN CONTINUE ON
      END
```

```
C     INS.COS
C
C       INS (INPUT SET), CALLED BY PATT, FILLS IN Y0 WITH
C  INITIAL VALUES.  WHEN USING DISK INPUT TO DRIVE PATT, SET IN=22 HERE.
C  FOR NO OUTPUT TO TERMINAL, CHANGE TYPE TO "CONTINUE!".

C  SET M2 BELOW 0 FOR FIRST ORDER PERTURB; 1 FOR 2ND ORDER (BOTH TERMS)

      SUBROUTINE INS(NOSP,YO,ISTAND)  !ISTAND=1 IF "STANDARD" DESIRED
      IMPLICIT DOUBLE PRECISION(A-H,O-Y)
      PARAMETER IN=5
      DIMENSION YO(1)
      COMMON /PATT1/ F,Q,S,SREC,W,DIFAL,DIFET,DIFRO
      COMMON /INS1/TOTL
C  SET ALPHA, ETA, RHO TO STEADY STATE VALUES
400   ALO=((1.-F-Q)+SQRT((Q+F-1.)**2+4*(F+1.)*Q))/(2*Q)
      ETO=1.0*(F*ALO/(1.+ALO))
      ROO=1.0*ALO
      TYPE 430,ALO,ETO,ROO
430   FORMAT(1X'ALPHA= 'E10.4,'  ETA= 'E10.4,'  RHO= 'E10.4)
      N=3*NOSP
      TYPE 460
460   FORMAT (' ENTER COEFF FOR AL,ET,RO')
      ACCEPT 480,ALC,ETC,ROC

C  SECOND ORDER TERMS ARE OF FORM A+B*COS(GL).
C  THIS SECTION COMPUTES THE A'S AND B'S.
C  FIRST PASS GIVES THE A'S; SECOND PASS THE B'S.
      XKK=0.
      PI=3.1415926
      XK=PI/TOTL       !  (TOTAL LENGTH)/PI = G
480   FORMAT (3D)
490   D1=(ALC*ETC+Q*ALC**2)/2.
      D2=(ALC*ETC)/2.
      C11=1.-ETO-2.*Q*ALO-DIFAL/S*4.*XKK**2
      C12=1-ALO
      C21=-(ETO)
      C22=-(1.+ALO+S*DIFET*4.*XKK**2)
      C23=F
      C31=1.
      C33=1.-DIFRO/W*4.*XKK**2
      E=(ETO+F/C33)/C22
C  HERE ARE B'S FOR X, Y, Z (ALPHA, ETA, RHO)
      BAL=(D1-(1.-ALO)*D2/C22)/(C11+(1.-ALO)*E)
      BRO=-BAL/C33
      BET=D2/C22+E*BAL
      IF(XKK.NE.0.) GO TO 600
C  HERE ARE A'S FOR X, Y, Z
      AAL=BAL
      AET=BET
```

```
      ARO=BRO
      XKK=XK
      GO TO 490
600   DXL=TOTL/(NOSP-1)
C     TYPE 500,BAL,AAL,BET,AET,BRO,ARO
500   FORMAT (1X,2E)

C  SET M2 BELOW 0 FOR FIRST ORDER PERTURB; 1 FOR 2ND ORDER (BOTH TERMS)
      M2=0
      HAFWAV=1.        !NUMBER OF HALF WAVELENGTHS IN CO1.
      DO 230 I=1,NOSP
      CO1=COS(1.*HAFWAV*XK*(I-1)*DXL)
      CO2=COS(2.*HAFWAV*XK*(I-1)*DXL)
      YO(3*I-2)=ALO+ALC*CO1+M2*(BAL*CO2+AAL)
      YO(3*I-1)=ETO+ETC*CO1+M2*(BET*CO2+AET)
      YO(3*I)=ROO+ROC*CO1+M2*(BRO*CO2+ARO)
230   CONTINUE
260   RETURN
      END
```

```
C      INS.PUL
C
C        INS (INPUT SET), CALLED BY PATT, FILLS IN Y0 WITH
C  INITIAL VALUES.  WHEN USING DISK INPUT TO DRIVE PATT, SET IN=22 HERE.
C  FOR NO OUTPUT TO TERMINAL, CHANGE TYPE TO "CONTINUE!".

       SUBROUTINE INS(NOSP,Y0,ISTAND)  !ISTAND=1 IF "STANDARD" DESIRED
       IMPLICIT DOUBLE PRECISION(A-H,O-Y)
       PARAMETER IN=5
       DIMENSION Y0(1)
       COMMON /PATT1/ FF,Q,S,SREC,W,DIFXOV,DIFYOV,DIFZOV
C  SET ALPHA, ETA, RHO TO STEADY STATE VALUES
400    AL=((1.-FF-Q)+SQRT((Q+FF-1.)**2+4*(FF+1.)*Q))/(2*Q)
       ET=1.0*(FF*AL/(1.+AL))
       RO=1.0*AL
       AL=1.0*AL
       TYPE 430,AL,ET,RO
430    FORMAT(1X'ALPHA= 'E10.4,'  ETA= 'E10.4,'  RHO= 'E10.4)
C  SET NORMAL LEVELS HERE
       AL=1.0317
       ET=32.4846
       RO=110
       N=3*NOSP
       DO 230 I=1,N-2,3
       Y0(I)=AL
       Y0(I+1)=ET
230    Y0(I+2)=RO
C  SET PULSE LEVELS HERE
       ALP=1180.216
       ETP=.05587
C  REMEMBER Y0 STORED X, Y, Z (ALPHA, ETA, RHO)
       DO 310 I=1,NOSP/10     !  PULSE WIDTH
       Y0(3*I-2)=ALP
       Y0(3*I-1)=ETP
310    CONTINUE
       RETURN
       END
```

```
C  PICTUR.FOR

C  PICTUR TAKES FOR23.DAT WHICH WAS CONSTRUCTED BY PATT AND SENDS EACH
C  SET OF INFORMATION TO PIC FOR PLOTTING.
C  YOU MUST SET THE DIMENSION STATEMENT TO APPROPRIATE SIZES, I.E.
C       YO(3*NOSP),ZYPIC(NOSP+2),ZDIS(NOSP+2).
C  PICTUR NEEDS PIC, PTCNEW & XONOUT.
      DOUBLE PRECISION YO,TOUT,DX
      DIMENSION YO(750),ZYPIC(252),ZDIS(252)
      COMMON /PIC2/ N,TOUT,IRUN          !3*NOSP, TIME OUT, RUN #
      COMMON /GO1/ DX
      CALL INITAL(5,254,11,1,0,0)       ! INITIALIZE PLOTTER
      CALL RSTR(0)                      ! CLEAR BUFFER
      ACCEPT 520,IDUMMY           !PERFORM DUMMY READ AFTER RSTR
520   FORMAT (G)
C  *  *  *  *  *  *  *  *  *
100   READ (23,500,END=800) N,TOUT,IRUN,DX   !READ FOR23 PARAMS
500   FORMAT(I,D,I,D)
      NPLT=N/3+2
      TYPE 510,NPLT
C  TELL USER HOW HE/SHE SHOULD HAVE SET DIMENSIONS IN THIS PROGRAM
510   FORMAT(1X'DIMENSION OF ZYPIC SHOULD HAVE BEEN'I5)
      DO 200 I=1,N
200   READ (23,530) YO(I)               !READ FOR23: YO
530   FORMAT(E10.4)      !FORM IS X1,Y1,Z1,X2,Y2,Z2,...AS PIC EXPECTS.
      CALL PIC(YO,NPLT,ZYPIC,ZDIS)        !CALL PIC
      GO TO 100                           !GET THE NEXT ONE
800   STOP
      END
```

```
C  PIC.FOR

C  PIC DOES THE PLOTTING.  IT NEEDS PTCNEW AND XONOUT.
C  PIC ALSO GIVES NUMBER VALUES FOR RIGHT AND LEFT ENDS
C  OF CONTAINER ON REQUEST
C  IF INPUT COMES FROM FILE, IN=22. OTHERWISE 5.
C  ALSO, IN THE FORMER CASE, YOU COULD CHANGE TYPE COMMAND
C  TO A "CONTINUE!".


      SUBROUTINE PIC(Y0,NPLT,ZYPIC,ZDIS)
      PARAMETER IN=5
      DOUBLE PRECISION Y0,TOUT,DX,XNUM,CHK
      DIMENSION Y0(1),LAB(4),ZYPIC(1),ZDIS(1),OR(4,2),XNUM(6)
C  PIC NEEDS THE DIMENSION NPLT PASSED TO IT.  PASSING THE Z ARRAYS
C  IN COMMON DOESN'T WORK

      COMMON /PIC2/ N,TOUT,IRUN    ! 3*NOSP, TIME OUT, RUN #
      COMMON /G01/ DX              ! DX=LENGTH/(NOSP-1)
      DATA LAB/'LOG XLOG YLOG ZDIST '/   ! AXIS LABELS
C     DATA LAB/'    X    Y    ZDIST '/   ! ALT AXIS LABELS
      DATA OR/0.,4.,0.,4.,6.5,6.5,1.,1./ ! X1,X2,X3,X4,Y1,Y2,Y3,Y4
      DATA NSCAL/1/                      ! GRAPH ORIGINS
      NOSP=NPLT-2                  ! NPLT IS DIM OF PLOTTING ARRAYS
      RNUM=IRUN
80    IF (NUMPLT.EQ.1) GO TO 600   ! BR IF NUMBERS WANTED
      TYPE 500
500   FORMAT (1X,'0=NO 1=AL 2=ET; 3=RO 4=ALL 5=SET 6=DMP
     1 7=AL-ET 8=AX')
C  5 LETS ONE CHOOSE NUMBERS.  6 DUMPS Y0 ONTO FOR23.DAT.
C  7 GIVES X-Y PHASE SPACE GRAPH. 8 LETS ONE CONTROL AXIS OF GRAPH.
      READ (IN,520,ERR=80) LOOK    ! TELLS WHAT TO DO
520   FORMAT (I)
      GO TO (100,100,100,200,300,700,800,900) LOOK
      RETURN                       ! IF 0 OR >8, RETURN

C  LOOK IS 8. EITHER LET PLOTTER SCALE AXIS, OR SET THEM YOURSELF.
900   TYPE 202
202   FORMAT (' 0=NO CHANGE; 1=SCALE; 2=FORCE')
      READ (5,30,ERR=900) NN
30    FORMAT (I)
      GO TO (80,612)NN+1
C  HERE IF USER SETS BOUNDARIES.
40    TYPE 628
628   FORMAT (' ENTER BOUNDARIES: X0,DELX,Y0,DELY')
      READ (5,90,ERR=40) AB,XDEL,ORD,YDEL
90    FORMAT (4D)
      NSCAL=0                 ! AXIS SCALE FLAG OFF.
      GO TO 80
C  HERE IF USER WANTS ROUTINE TO SCALE AXES.
```

```
612   NSCAL=1                  ! AXES SCALED. FLAG ON.
      GO TO 80

C  LOOK IS 6. DUMP Y0 TO DISK.
700   WRITE (23,710) (Y0(K),K=1,N)
710   FORMAT (1X,3D)
      RETURN

C  LOOK IS 5. CHOOSE PLOTS OR NUMBERS.
300   TYPE 510
510   FORMAT (1X,'0=PLOT; 1=NUMBERS ',$)
      READ (5,520,ERR=300)NUMPLT
      GO TO 80
600   TYPE 530
C30   FORMAT (' 0=ALPH(1); 1=ALL(1); 2=ALPH(1&LAST); 3=ALL(1&LAST);
C     1 5=SET'$)
530   FORMAT (' GO ',$)
      READ (5,520,ERR=600) LOOK
C  WHEN GIVING NUMBERS, LET USER SELECT WHAT TO SEE.
      J=3
      K=2
      L=0
      GO TO (605,610,615,615,300)LOOK
      J=1
      K=1
      GO TO 615
605   K=1
      GO TO 615
610   J=1
615   DO 625 I1=1,J
      IPNT=1
      DO 620 I2=1,K
      L=L+1
      XNUM(L)=DLOG10(Y0((IPNT-1)*3+I1))
      IPNT=NOSP
620   CONTINUE
625   CONTINUE
      IF (J*K.EQ.2) GO TO 635    ! BR TO PRINT LOTS OF DECIMAL PLACES.
      TYPE 550,(XNUM(I),I=1,J*K)
550   FORMAT (1X,6F12.5)
      RETURN
635   TYPE 555,XNUM(1),XNUM(2)
555   FORMAT (1X,2F15.9)
      RETURN

C  LOOK IS 7; PHASE SPACE GRAPH. NEGATIVE CONCENTRATION
C  REPLACED BY PREVIOUS VALUE FOR SAKE OF GRAPH. Y0 IS NOT
C  CHANGED THOUGH.
800   NCNT=0                   !NOTE NEGATIVE CONCENTRATIONS
      DO 151 I=1,NOSP
```

```
      CHK=YO((I-1)*3+1)
      IF (CHK.GE.0.) GO TO 141
      NCNT=NCNT+1
      ZDIS(I)=ZDIS(I-1)+(-1)**NCNT*.05
      GO TO 151
141   ZDIS(I)=DLOG10(CHK)
C141  ZDIS(I)=CHK        !!!!!  USE THIS IF YOU DON'T WANT LOGS.
151   CONTINUE
      LOOK=2              !X-AXIS IS SET NOW. PUT ETA ON Y AXIS.
      NPH=1               ! PHASE SPACE FLAG.
      IF (NCNT.EQ.0) GO TO 110  !BR IF NO NEG CONCENTRATIONS.
      TYPE 540,NCNT
      GO TO 110

C  LOOK IS 4. PRINT X, Y AND Z GRAPHS.
200   DO 400 J=1,3                ! IF LOOK=4,DO 3 PLOTS
      LOOK=J

C  LOOK IS 1,2 OR 3 IF BRANCHED TO HERE.
100   ZDIS(1)=0.
      NPH=4
      DO 50 I=1,NOSP-1            ! CREATE X COORDINATES
50    ZDIS(I+1)=ZDIS(I)+DX        !    (DISTANCE)
110   ICNT=ICNT+1                 !CHECK IF PLOTTER PAGE IS FULL.
      IF (ICNT.LT.5) GO TO 220
      CALL RSTR(2)                          !NEW PAGE
      ACCEPT 520,IDUMMY   !NEED A DUMMY READ AFTER A RSTR.
      ICNT=1
220   CALL PLOT (OR(ICNT,1),OR(ICNT,2),-3)   !SET PLOT ORIGIN.
C  CREATE Y COORD FOR APPROPRIATE GRAPH. YO IS STORED X1,Y1,Z1,X2,Y2,...
      NCNT=0                    !NOTE NEGATIVE CONCENTRATIONS
      DO 150 I=1,NOSP
      CHK=YO((I-1)*3+LOOK)
      IF (CHK.GE.0.) GO TO 140
      NCNT=NCNT+1
      ZYPIC(I)=ZYPIC(I-1)+(-1)**NCNT*.05
      GO TO 150
140   ZYPIC(I)=DLOG10(CHK)
C140  ZYPIC(I)=CHK        !!!!!  USE THIS IF YOU DON'T WANT LOGS.
150   CONTINUE
      IF (NCNT.EQ.0) GO TO 158
      TYPE 540,NCNT
540   FORMAT (1X,I,'  NEG CONCENTRATION POINTS ON NEXT GRAPH.')
158   ZDIS(NOSP+1)=AB     !SET AXIS PARAMETERS IN CASE SCALE NOT USED
      ZDIS(NOSP+2)=XDEL
      ZYPIC(NOSP+1)=ORD
      ZYPIC(NOSP+2)=YDEL
      IF (NSCAL.EQ.1) GO TO 163     ! BR IF SCALE
      IF (NPH.EQ.1) GO TO 164       ! BR IF NO SCALE AND PHASE SPACE.
      GO TO 160
```

```
163   CALL SCALE (ZYPIC,3.,NOSP,1)
160   CALL SCALE (ZDIS,3.,NOSP,1) ! 3 INCH X AXIS.

C  PLOT

164   CALL AXIS (0.,0.,LAB(LOOK),5,3.,90.0,ZYPIC(NOSP+1),ZYPIC(NOSP+2)
      1,2)                                  ! 3 INCH Y AXIS
      CALL AXIS (0.,0.,LAB(NPH),-5,3.,0.,ZDIS(NOSP+1),
      1ZDIS(NOSP+2),2)              !3 INCH X AXIS
      CALL LINE (ZDIS,ZYPIC,NOSP,1,0,0)     ! DRAW GRAPH.
C     CALL SYMBOL(1.5,3.5,.14,'RUN =',0.,5) ! ADD HEADER LABELS
C     CALL NUMBER (999.,999.,.14,RNUM,0.,-1)
      CALL SYMBOL (1.5,3.,.14,'TAU =',0.,5)
      CALL NUMBER (999.,999.,.14,TOUT,0.,2)
390   XRST=-OR(ICNT,1)  ! GO TO LOWER LEFT CORNER OF PLOT PAGE.
      YRST=-OR(ICNT,2)
      CALL PLOT(XRST,YRST,-3)
      CALL RSTR(0)
      ACCEPT 520,IDUMMY       ! NECESSARY DUMMY READ.
400   CONTINUE
      GO TO 80
      END
```

```
ENTRY RUNTIM
SEARCH MONSYM
       P=17
RUNTIM:MOVEI 1,-5
       PUSH P,2
       PUSH P,3
       RUNTM
       POP P,3
       POP P,2
       MOVE 0,1
       POPJ 17,0
       END
```

```
C     WIND.FOR

C     THIS IS A MAINLINE PROGRAM WHICH UTILIZES A DOUBLE PRECISION
C     VARIANT OF ZEROS.FOR TO FIND THE ROOTS OF THE EQUATION
C     G(X)=0.  WIND MAY NEED PLOTTING PROGRAMS XONOUT AND PTCNEW.

      IMPLICIT DOUBLE PRECISION (A-H,O-Y)
      PARAMETER NOSP=250, NOSPX=252,NOSP2=500
      DIMENSION RT(100),ZX(NOSPX),ZY(NOSPX),ZAE(502)
      COMMON Q,DUM2,ROSET,S,W,F,NFCN

C  INITIALIZE AND GET PARAMETERS FROM FOR21.DAT.
      TYPE 50
50    FORMAT (' ENTER 1 IF PLOTTER ON LINE  ',$)
      ACCEPT 30,NN
      IF (NN.EQ.0) GO TO 62
      CALL INITAL(5,254,11,1,0,0)
      CALL RSTR(0)
      READ (5,30)IDUMMY
62    NFCN=1
      NFRST=0
      REWIND 21                        !(FOR21.DAT SET BY PARSET)
      READ (21,105,ERR=460) A,B,HPL,VK1,VK2,VK3,VK4,VK5,F,H0,
      1  EPS,MF,DIFX,DIFY,DIFZ          !H0, EPS, MF ARE GEAR PARAMS
105   FORMAT(11D,I,3D)
      VK1=VK1*HPL*HPL           !SET CONSTANTS FOR DIF. EQ.'S
      VK2=VK2*HPL              !DIFFUN & PDB GET THESE THRU COMMON PATT1
      VK3=VK3*HPL
      Q=2.*VK1*VK4*A/(VK2*VK3*B)
      S=DSQRT(VK3*B/(VK1*A))
      SREC=1./S
      CON=DSQRT(VK1*VK3*A*B)
      W=VK5/CON
      DIFAL=100.*DIFX/CON
      DIFET=100.*DIFY/CON
      DIFRO=100.*DIFZ/CON

C  FUNCTION 1:  FOR A VALUE OF Z GIVEN BY USER, FUNCTION 1 GIVES
C               COORDINATES OF INTERSECTIONS OF THE TWO TRADE LINES.
C  FUNCTION 2:  FUNCTION 2 TELLS FOR WHAT Z VALUE THE NUMBER OF
C               INTERSECTION POINTS OF THE TRADE LINES CHANGES FROM
C               3 TO 1, OR VICE VERSA. THEIR IS SOMETIMES AN
C               EXTRANEOUS ROOT.
C    RANGE AND NUMBER OF ITERATIONS FOR FUNC 1 & 2 HAVE DEFAULT VALUES
C  GRAPH        GRAPHS TRADE LINES IN X-Y PHASE SPACE.
C  LOGGR        SAME GRAPH BUT LOG SCALES ON AXES.
180   TYPE 190
190   FORMAT(/,' 1 FOR FUNC 1, 2 FOR FUNC 2, 3 FOR GRAPH, 4 FOR LOGGR')
      ACCEPT 30,NFCN
```

```
30    FORMAT (I)
      GO TO (70,70,600,600),NFCN
      GO TO 180

C  FUNCTIONS 1 & 2
70    TYPE 80
80    FORMAT(' ENTER XSTART,XSTOP')    ! SEE ZEROS BELOW.
      ACCEPT 90,XSTART,XSTOP
      IF (XSTART.NE.XSTOP) GO TO 100
      XSTART=1.000001                  ! DEFAULTS
      XSTOP=1./Q
90    FORMAT (4D)
100   TYPE 110
110   FORMAT(' ENTER NTOT,NSQUE')      ! SEE ZEROS BELOW
      ACCEPT 120,NTOT,NSQUE
120   FORMAT(2I)
      IF (NTOT.NE.0) GO TO 122
      NTOT=1000                        ! DEFAULTS
      NSQUE=100
122   IF (NFCN.EQ.2) GO TO 137

C  FUNCTION 1.  NODE COORDINATES FOR A GIVEN Z.
      TYPE 125
125   FORMAT (' ENTER RO SET = ',$)
      ACCEPT 90, ROSET
128   CALL ZEROS(XSTART,XSTOP,NTOT,NSQUE,RT,NRT)
      IF (NRT.NE.0) GO TO 145
129   TYPE 130
130   FORMAT(/,' NO ROOTS FOUND',/)
      GO TO 180

C  FUNCTION 2.  IF YOU CHANGE F HERE, IT STAYS CHANGED FOR REST OF PROG
C  EQUATION FOR # OF ROOTS OF A CUBIC OUT OF
C  C.R.C. STANDARD MATHEMATICAL TABLES.
137   TYPE 138
138   FORMAT (' NEW F=  ',$)
      ACCEPT 90,TMP
      IF (TMP.NE.0.) F=TMP
      CALL ZEROS(XSTART,XSTOP,NTOT,NSQUE,RT,NRT)
      IF (NRT.EQ.0) GO TO 129
      TYPE 155
155   FORMAT (/,' ROOT #   RHO VALUE',/)
      DO 165 N=1,NRT
165   TYPE 160,N,RT(N)
      GO TO 180
145   TYPE 150
150   FORMAT(/,' ROOT #   ALPHA VALUE            ETA VALUE',/)
      DO 170 N=1,NRT
      ETA=F*ROSET/(1.+RT(N))
      TYPE 160,N,RT(N),ETA
```

```
160   FORMAT(3X,I3,3X,2(1PE22.15,1X))
170   CONTINUE
      GO TO 180

C  GRAPHS.
600   TYPE 620
620   FORMAT (' ENTER L & R SIDES')
      ACCEPT 90,ZX(1),ZX(NOSP)
      IF (ZX(1).NE.ZX(NOSP)) GO TO 630
      ZX(1)=1.01           !DEFAULTS
      ZX(NOSP)=(.95/Q)
630   TYPE 125
      ACCEPT 90, TMP
      IF (TMP.EQ.0.) GO TO 632
      ROSET=TMP            !DEFAULT
632   IF (NFCN.EQ.3) GO TO 635   ! BR IF NOT LOG GRAPH.
      ZX(1)=ALOG10(ZX(1))
      ZX(NOSP)=ALOG10(ZX(NOSP))
635   DELF=(ZX(NOSP)-ZX(1))/(NOSP-1)   !DELTA FOR X AXIS
      DO 605 I=1,NOSP-1
605   ZX(I+1)=ZX(I)+DELF
      DO 610 I=1,NOSP
      ZXX=ZX(I)
      IF (NFCN.EQ.3) GO TO 607
C  ZAE(1) TO (250) HAS Y TRADE LINE.  251 TO 500 IS X TRADE.
C  BR IF NORMAL GRAPH.

C  CONTINUE IF LOG GRAPH
C  GET X VALUE FROM LOGX, FIND Y AND TAKE ITS LOG.
      ZXX=10.**ZXX
      ZAE(I)=ALOG10(F*ROSET/(1.+ZXX))
      TMP=((ZXX-Q*ZXX*ZXX)/(ZXX-1.))
      IF (TMP.GT.0.) GO TO 608
      ZAE(NOSP+I)=ZAE(NOSP+I-1)
      GO TO 610
608   ZAE(NOSP+I)=ALOG10(TMP)
      GO TO 610
607   ZAE(I)=(F*ROSET/(1.+ZXX))
      ZAE(NOSP+I)=((ZXX-Q*ZXX*ZXX)/(ZXX-1.))
610   CONTINUE

C  PLOT
      SIZE=3.          !SIZE OF GRAPH AXES IN INCHES.
      CALL PLOT(0.,0.,-3)       ! ORIGIN
      TYPE 202
C  LET PROGRAM SCALE AXES, OR LET USER SET AXES SCALE.
202   FORMAT (' 0=SAME AXIS AS BEFORE; 1=SCALE; 2=FORCE')
      ACCEPT 30,NN
      NFRST=NFRST+1
      GO TO (612,627) NN
```

```
      IF (NFRST.EQ.1) GO TO 612   !IF VERY FIRST GRAPH, BR & SCALE.
      GO TO 615             ! IF NN=0, DON'T CHANGE AXES PARAMS.
627   TYPE 628
C  HERE IF USER ENTERS AXES PARAMS.
628   FORMAT (' ENTER BOUNDARIES: X0,DELX,Y0,DELY') !DELX IS DELTA X.
      ACCEPT 90,ZX(NOSP+1),ZX(NOSPX),ZAE(NOSP2+1),ZAE(502)
      GO TO 615
C  HERE IF SCALE.
612   CALL SCALE(ZX,SIZE,NOSP,1)
      CALL SCALE(ZAE,SIZE,NOSP2,1)
C  ZAE HAS BOTH TRADE LINES IN IT FOR THIS SCALING.
C  ZY WILL HAVE CURVE FOR PLOT.
615   ZY(NOSP+1)=ZAE(NOSP2+1)
      ZY(NOSPX)=ZAE(NOSP2+2)
      IF (NFCN.EQ.3) GO TO 655
      CALL AXIS(0.,0.,'LOG X',-5,SIZE,0.,ZX(NOSP+1),ZX(NOSPX),2)
      CALL AXIS (0.,0.,'LOG Y',5,SIZE,90.0,ZY(NOSP+1),ZY(NOSPX),2)
      GO TO 658
655   CALL AXIS(0.,0.,'X',-1,SIZE,0.,ZX(NOSP+1),ZX(NOSPX),2)
      CALL AXIS (0.,0.,'Y',1,SIZE,90.0,ZY(NOSP+1),ZY(NOSPX),2)

658   DO 650 I=0,1          ! PLOT BOTH TRADE LINES.
      DO 645 J=1,NOSP
645   ZY(J)=ZAE(I*NOSP+J)
      LIN=0*I
      CALL LINE (ZX,ZY,NOSP,1,LIN,0)
650   CONTINUE
      CALL RSTR(2)
      READ (5,30) IDUMMY
      GO TO 180
460   PAUSE  'WHERE IS FILE?'
      END

      FUNCTION G(X)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON Q,DUM2,ROSET,S,W,F,NFCN
      GO TO (10,20,30,40,50,60,70,80),NFCN
C  IF TRADE LINES INTERSECT, G IS ZERO
10    G=Q*X**3-(1.-Q)*X**2+(F*ROSET-1)*X-F*ROSET
      RETURN
C  G ZERO AT POINT WHERE TRADE LINES JUST TOUCH.
C  SEE CRC STANDARD MATH TABLES FOR # OF ROOTS FOR A CUBIC.
20    P=(Q-1.)/Q
      QQ=(F*X-1.)/Q
      R=-F*X/Q
      A=1./3.*(3.*QQ-P**2)
      B=1./27.*(2.*P**3-9.*P*QQ+27.*R)
      G=(B*1.E-6)**2/4.+(A*1.E-4)**3/27.
      RETURN
30    G=0.
```

```
      RETURN
40    G=0.
      RETURN
50    G=0.
      RETURN
60    G=0.
      RETURN
70    G=0.
      RETURN
80    G=0.
      RETURN
      END

C     ZEROS.FOR    COMPLIMENTS OF RICHARD J. HAYDEN.

C     THIS SUBROUTINE FINDS THE ROOTS OF THE EQUATION G(X)=0 (THE
C     ZEROS OF G(X)) WHICH ARE LOCATED AT OR BETWEEN XSTART AND
C     XSTOP. THE FUNCTION G(X) MUST BE WRITTEN IN A SEPARATE
C     FUNCTION SUBROUTINE. THE MAINLINE PROGRAM SENDS TO ZEROS THE
C     FOLLOWING QUANTITIES:
C             XSTART=START LIMIT OF INVESTIGATION RANGE
C             XSTOP=STOP LIMIT OF INVESTIGATION RANGE
C             NTOT=NUMBER OF DIVISIONS INTO WHICH THE INVESTIGATION
C                  RANGE IS INITIALLY DIVIDED
C             NSQUE=NUMBER OF SQUEEZES BY A FACTOR OF TEN AFTER
C                   A ROOT IS ROUGHLY LOCATED
C     THE SUBROUTINE RETURNS TO THE MAINLINE PROGRAM THE QUANTITIES:
C             NRT=NUMBER OF ROOTS FOUND
C             RT(N)  (N=1,2,---NRT) THESE ARE THE NRT ROOTS FOUND
C                  BY THE SUBROUTINE. RT(N) MUST BE DIMENSIONED IN
C                  THE MAINLINE PROGRAM.

      SUBROUTINE ZEROS(XSTART,XSTOP,NTOT,NSQUE,RT,NRT)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION RT(1)
      A=XSTART
      B=XSTOP
      FNTOT=NTOT
      NRT=0
      DO 310 J=1,NTOT
      FJ=J
      X=A+(FJ-1.)*(B-A)/FNTOT
      Y=A+FJ*(B-A)/FNTOT
      IF (G(X)) 110,250,110
110   IF (G(X)*G(Y)) 120,310,310
120   DO 220 L=1,NSQUE
      DO 190 K=1,10
      FK=K
      U=X+(FK-1.)*(Y-X)/10.
      V=X+FK*(Y-X)/10.
```

```
      IF (G(U)) 180,270,180
180   IF (G(U)*G(V)) 200,190,190
190   CONTINUE
200   X=U
      Y=V
220   CONTINUE
      S=(X+Y)/2.
C     TEST TO THROW OUT INFINITIES
      IF (G(S)*G(X)) 233,236,234
233   IF (G(S)*G(Y)) 310,238,235
234   IF ((G(S)-G(X))/G(X)) 236,236,310
235   IF ((G(Y)-G(S))/G(Y)) 310,236,236
236   R=S
      GO TO 290
238   R=Y
      GO TO 290
250   R=X
      GO TO 290
270   R=U
      GO TO 290
290   NRT=NRT+1
      RT(NRT)=R
310   CONTINUE
      IF (G(B)) 350,312,350
312   NRT=NRT+1
      RT(NRT)=B
350   RETURN
      END
```

```
C     REG.FOR

C  THIS PROGRAM FOR THE OREGONATOR WITHOUT DIFFUSION DRAWS
C  THE REGIONS IN F - K(5) SPACE WHERE THE STEADY-STATE
C  SOLUTION IS STABLE AND UNSTABLE.
C  THE EQUATIONS USED ARE DEVELOPED BY MURRAY (SEE REFERENCES).
C  J. CHEM PHYS, V. 61, NO. 9, NOV 1 1974, PP 3610-3613.


      IMPLICIT DOUBLE PRECISION (A-H,O-Y)
      DIMENSION RT(100),ZFAXIS(252),ZKAXIS(252)
      COMMON Q,AL,ET,S,W,DUM2,NFCN

C  INITIALIZE AND GET PARAMS AND COMPUTE CONSTANTS.
      CALL INITAL(5,254,11,1,0,0)
      CALL RSTR(0)
      READ (5,30)IDUMMY
      NFCN=1
      REWIND 21                          !(FOR21.DAT SET BY PARSET)
      READ (21,105,ERR=460) A,B,HPL,VK1,VK2,VK3,VK4,VK5,DUM,HO,
      1  EPS,MF,DIFX,DIFY,DIFZ          !HO, EPS, MF ARE GEAR PARAMS
105   FORMAT(11D,I,3D)
      VK1=VK1*HPL*HPL           !SET CONSTANTS FOR DIF. EQ.'S
      VK2=VK2*HPL              !DIFFUN & PDB GET THESE THRU COMMON PATT1
      VK3=VK3*HPL
      Q=2.*VK1*VK4*A/(VK2*VK3*B)
      S=DSQRT(VK3*B/(VK1*A))
      SREC=1./S
      CON=DSQRT(VK1*VK3*A*B)
      W=VK5/CON
      DIFAL=100.*DIFX/CON
      DIFET=100.*DIFY/CON
      DIFRO=100.*DIFZ/CON

C  FUNCTION 1, EQ 20 IN MURRAY'S PAPER, GIVES EXTREME VALUES
C  OF F FOR CURVE.

C  FUNCTION 2, EQ 22 IN MURRAY, GIVES F VALUES FOR THE VALUE
C  OF K(5) IN PARAMS.

70    TYPE 80
80    FORMAT(' ENTER XSTART,XSTOP')      ! SEE ZEROS BELOW.
      ACCEPT 90,XSTART,XSTOP
90    FORMAT (2D)
100   TYPE 110
110   FORMAT(' ENTER NTOT,NSQUE')        ! SEE ZEROS BELOW.
      ACCEPT 120,NTOT,NSQUE
120   FORMAT(2I)
      CALL ZEROS(XSTART,XSTOP,NTOT,NSQUE,RT,NRT)
```

```
      IF (NRT.NE.0) GO TO 140
      TYPE 130
130   FORMAT(/,' NO ROOTS FOUND',/)
      GO TO 180
140   IF (NFCN.NE.1) GO TO 145
      ZFAXIS(1)=RT(1)           !SAVE FOR GRAPH.
      ZFAXIS(250)=RT(2)
145   TYPE 150
150   FORMAT(/,' ROOT #   ROOT VALUE',/)
      DO 170 N=1,NRT
      TYPE 160,N,RT(N)
160   FORMAT(3X,I3,3X,1PE22.15)
170   CONTINUE
180   TYPE 190
190   FORMAT(/,' 1 FOR FUNC 1, 2 FOR FUNC 2, 3 FOR GRAPH')
      ACCEPT 30,NFCN
30    FORMAT (I)
      GO TO (70,70,600),NFCN
      GO TO 180
      STOP

C  HERE FOR GRAPH.  THE BOUNDS OF THE GRAPH HAVE ALREADY BEEN SET
C  BY THE INITIAL AUTO RUN THROUGH FOR FUNCTION 1.
600   DELF=(ZFAXIS(250)-ZFAXIS(1))/249.
      DO 605 I=1,249
605   ZFAXIS(I+1)=ZFAXIS(I)+DELF
      DO 610 I=1,250
C  THE FOLLOWING COMPUTATION FOR W IS EQ 22 IN MURRAY'S PAPER.
      F=ZFAXIS(I)
      AL=((1.D0-F-Q)+DSQRT((Q+F-1.D0)**2+4.D0*(F+1.D0)*Q))/(2.D0*Q)
      ET=(F*AL/(1.+AL))
      RO=AL
      E=S*ET+(1./S+2.*Q*S)*AL+1./S-S
      W=-.5/E*(E**2+F*(1.-AL))+.5/E*SQRT((E**2+F*(1.-AL))**2
     1 -4.*E**2*(2.*Q*AL**2+AL*(Q-1.)+F))
C  PLOT
610   ZKAXIS(I)=SQRT(VK1*VK3*A*B)*W
      CALL SCALE(ZFAXIS,6.,250,1)
      CALL PLOT(0.,0.,-3)
      CALL AXIS(0.,0.,'F',-1,5.,0.,ZFAXIS(251),ZFAXIS(252),2)
      CALL SCALE (ZKAXIS,5.,250,1)
      CALL AXIS (0.,0.,'K5',2,5.,90.0,ZKAXIS(251),ZKAXIS(252),2)
      CALL LINE (ZFAXIS,ZKAXIS,250,1,0,0)
      CALL RSTR(2)
      READ (5,30) IDUMMY      !DUMMY READ.
      GO TO 180
460   PAUSE  'WHERE IS FILE?'
      END

      FUNCTION G(F)
```

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON Q,AL,ET,S,W,DUM2,NFCN
      GO TO (10,20,30,40,50,60,70,80),NFCN
C  EQ. 20 IN MURRAY FINDS EXTREMES OF F.
10    G=2.*Q*(2.+3.*F)-(2.*F+Q-1.)*((1.-F-Q)
      1 +SQRT((1.-F-Q)*(1.-F-Q)+4.*Q*(1.+F)))
      RETURN
C  MAKES USE OF EQ 22 IN MURRAY TO FIND AN F GIVEN A W (I.E., K(5)).
20    AL=((1.D0-F-Q)+DSQRT((Q+F-1.D0)**2+4.D0*(F+1.D0)*Q))/(2.D0*Q)
      ET=(F*AL/(1.+AL))
      RO=AL
      E=S*ET+(1./S+2.*Q*S)*AL+1./S-S
      G=-W-.5/E*(E**2+F*(1.-AL))+.5/E*SQRT((E**2+F*(1.-AL))**2
      1 -4.*E**2*(2.*Q*AL**2+AL*(Q-1.)+F))
      RETURN
30    G=0.
      RETURN
40    G=0.
      RETURN
50    G=0.
      RETURN
60    G=0.
      RETURN
70    G=0.
      RETURN
80    G=0.
      RETURN
      END

C     ZEROS.F4        !COMPLIMENTS OF RICHARD J. HAYDEN.

C     THIS SUBROUTINE FINDS THE ROOTS OF THE EQUATION G(X)=0 (THE
C     ZEROS OF G(X)) WHICH ARE LOCATED AT OR BETWEEN XSTART AND
C     XSTOP. THE FUNCTION G(X) MUST BE WRITTEN IN A SEPARATE
C     FUNCTION SUBROUTINE. THE MAINLINE PROGRAM SENDS TO ZEROS THE
C     FOLLOWING QUANTITIES:
C             XSTART=START LIMIT OF INVESTIGATION RANGE
C             XSTOP=STOP LIMIT OF INVESTIGATION RANGE
C             NTOT=NUMBER OF DIVISIONS INTO WHICH THE INVESTIGATION
C                  RANGE IS INITIALLY DIVIDED
C             NSQUE=NUMBER OF SQUEEZES BY A FACTOR OF TEN AFTER
C                   A ROOT IS ROUGHLY LOCATED
C     THE SUBROUTINE RETURNS TO THE MAINLINE PROGRAM THE QUANTITIES:
C             NRT=NUMBER OF ROOTS FOUND
C             RT(N)  (N=1,2,---NRT) THESE ARE THE NRT ROOTS FOUND
C                  BY THE SUBROUTINE. RT(N) MUST BE DIMENSIONED IN
C                  THE MAINLINE PROGRAM.

      SUBROUTINE ZEROS(XSTART,XSTOP,NTOT,NSQUE,RT,NRT)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
```

```
      DIMENSION RT(1)
      A=XSTART
      B=XSTOP
      FNTOT=NTOT
      NRT=0
      DO 310 J=1,NTOT
      FJ=J
      X=A+(FJ-1.)*(B-A)/FNTOT
      Y=A+FJ*(B-A)/FNTOT
      IF (G(X)) 110,250,110
110   IF (G(X)*G(Y)) 120,310,310
120   DO 220 L=1,NSQUE
      DO 190 K=1,10
      FK=K
      U=X+(FK-1.)*(Y-X)/10.
      V=X+FK*(Y-X)/10.
      IF (G(U)) 180,270,180
180   IF (G(U)*G(V)) 200,190,190
190   CONTINUE
200   X=U
      Y=V
220   CONTINUE
      S=(X+Y)/2.
C     TEST TO THROW OUT INFINITIES
      IF (G(S)*G(X)) 233,236,234
233   IF (G(S)*G(Y)) 310,238,235
234   IF ((G(S)-G(X))/G(X)) 236,236,310
235   IF ((G(Y)-G(S))/G(Y)) 310,236,236
236   R=S
      GO TO 290
238   R=Y
      GO TO 290
250   R=X
      GO TO 290
270   R=U
      GO TO 290
290   NRT=NRT+1
      RT(NRT)=R
310   CONTINUE
      IF (G(B)) 350,312,350
312   NRT=NRT+1
      RT(NRT)=B
350   RETURN
      END
```

```
C  SMALL.FOR

C  SMALL NEEDS LIB:IMSL/LIB PROGRAMS TOO.
C  SMALL SOLVES EIGENVALUE PROBLEM TO FIND GENERAL SOLUTION TO
C  EQUATIONS, GIVING EIGENVALUE
C  AND ASSOCIATED EIGENVECTORS.  IT THEN MAKES 6 X 6 BOUNDARY
C  CONDITION MATRIX, AND REPORTS ON DET VALUE FOR GIVEN LENGTHS.
C  SOMETIMES THERE ARE CRITICAL LENGTHS FOR WHICH DET = 0.
C  USER THEN SUPPLIES THIS CRITICAL LENGTH AS INPUT AND PROGRAM
C  SOLVES 6X6 BOUNDARY MATRIX, GIVING RATIO OF COEFFICIENTS.
C  ONCE FIRST-ORDER SOLUTIONS WERE FOUND TO BE A SIMPLE COS,
C  SMALL WAS REPLACED BY SMALLK.
      IMPLICIT DOUBLE PRECISION(A-B,D-H,O-Y)
      IMPLICIT COMPLEX(C)
      COMPLEX A(10,10),B(10,10),EIGA(10),EIGB(10),WK(10,20)
      DIMENSION LAB(13),CFIN(6),CTMP(6)
      COMMON CLAM(3),CZ(10,10),CK(6)
      COMMON /CD/ C(6,6),IORD(6)
      DATA LAB/'  S    Q    W   SREC AL   ET   RO   ADOT EDOT RDOT
      1 DFAL DFET DFRO'/

C  INITIALIZE AND GET PARAMS AND COMPUTE CONSTANTS.
      REWIND 21                        !(FOR21.DAT SET BY PARSET)
      READ (21,410,ERR=460) AA,BB,HPL,VK1,VK2,VK3,VK4,VK5,F,H0,
      1  EPS,MF,DIFX,DIFY,DIFZ         !H0, EPS, MF ARE GEAR PARAMS
410   FORMAT(11D,I,3D)
      VK1=VK1*HPL*HPL
      VK2=VK2*HPL
      VK3=VK3*HPL
      Q=2.*VK1*VK4*AA/(VK2*VK3*BB)
      S=DSQRT(VK3*BB/(VK1*AA))
      SREC=1./S
      TCON=DSQRT(VK1*VK3*AA*BB)
      W=VK5/TCON
      DIFAL=100.*DIFX/TCON
      DIFET=100.*DIFY/TCON
      DIFRO=100.*DIFZ/TCON
      TYPE 500,LAB(1),S
      TYPE 500,LAB(2),Q
      TYPE 500,LAB(3),W
      TYPE 500,LAB(4),SREC
      TYPE 500,LAB(11),DIFAL
      TYPE 500,LAB(12),DIFET
      TYPE 500,LAB(13),DIFRO
400   AL=((1.D0-F-Q)+DSQRT((Q+F-1.D0)**2+4.D0*(F+1.D0)*Q))/(2.D0*Q)
      ET=(F*AL/(1.+AL))
      RO=AL
450   TYPE 500,LAB(5),AL
      TYPE 500,LAB(6),ET
```

```
      TYPE 500,LAB(7),RO
500   FORMAT(1X,A5,' = ',1PD)
      GO TO 480
460   PAUSE 'WHERE IS FILE'
      STOP

C  THE NEXT SECTION USES IMSL ROUTINE EIGZC TO SOLVE THE
C  EIGENVALUE MATRIX PROBLEM.  IT PRINTS THE EIGENVALUES AND VECTORS.
C  AX = BX

480   IJOB=2
      IA=10
      IB=10
      IZ=10
      NMAT=3
      DO 130 I=1,NMAT
      DO 120 J=1,NMAT
      B(I,J)=CMPLX(0.,0.)
120   CONTINUE
130   CONTINUE
      B(1,1)=DIFAL/S
      B(2,2)=DIFET*S
      B(3,3)=DIFRO/W
      A(1,1)=-ET-2.*Q*AL+1.
      A(1,2)=1.-AL
      A(1,3)=0.
      A(2,1)=-ET
      A(2,2)=-(AL+1.)
      A(2,3)=F
      A(3,1)=1.
      A(3,2)=0.
      A(3,3)=-1.
      CALL EIGZC(A,IA,B,IB,NMAT,IJOB,EIGA,EIGB,CZ,IZ,WK,INFER,IER)
      DO 220 J=1,NMAT
      CLAM(J)=EIGA(J)/EIGB(J)
      CK(2*J-1)=CSQRT(CLAM(J))
      CK(2*J)=-CK(2*J-1)
      RLAM=REAL(CLAM(J))
      AILAM=AIMAG(CLAM(J))
      TYPE 180,J,RLAM,AILAM
180   FORMAT(/,' LAMBDA(',I2,')=(',1PE14.7,',',1PE14.7,')')
      DO 185 L=0,1
      K=2*J-1+L
      RLK=REAL(CK(K))
      AIK=AIMAG(CK(K))
185   TYPE 187,K,RLK,AIK
187   FORMAT('     K(',I2,')=(',1PE14.7,',',1PE14.7,')')
      TYPE 190,J
190   FORMAT(' EIGENFUNCTION #',I2)
      DO 210 I=1,NMAT
```

```
      RX=REAL(CZ(I,J))
      AIX=AIMAG(CZ(I,J))
      TYPE 200,I,RX,AIX
200   FORMAT(' X(',I2,')=(',1PE14.7,',',1PE14.7,')')
210   CONTINUE
220   CONTINUE
      TYPE 230,INFER
230   FORMAT(' INFER=',I5)
      P=REAL(WK(1,1))
      TYPE 235,P
235   FORMAT(' PERFORMANCE INDEX=',1PE14.7)  !SEE EIGZC.

C  NEXT PRINT OUT VALUE OF DETERMINANT FOR A GIVEN LENGTH.

10    TYPE 20
20    FORMAT(' L=',$)
      ACCEPT 30,X
30    FORMAT(E)
      IF(X.FQ.83.) STOP      !83 MEANS STOP
      IF (X.EQ.85) GO TO 31  !85 MEANS GO FIND C VALUES
      CY=CG(X)
      YREAL=REAL(CY)
      YIMAG=AIMAG(CY)
      TYPE 40,YREAL,YIMAG
40    FORMAT(' DETERMINANT VALUE=',1PE14.7,',',1PE14.7,/)
C     GO TO 10     !LEAVE THIS BR ACTIVE UNTIL THE 0 DETERMINANT
C                     IS FOUND.  THEN KILL THE BRANCH, PUT IN PROPER
C                     LENGTH AND PRINT OUT 6 X 6  MATRIX MATRIX WHICH
C                     DESCRIBES BOUNDARY CONDITIONS.
C                     DIAGONALIZED BY CDET.
      DO 55 J=1,6
      DO 54 I=1,6
      YREAL=REAL(C(I,J))
      YIMAG=AIMAG(C(I,J))
      TYPE 57,I,J,YREAL,YIMAG
54    CONTINUE
57    FORMAT(' C(',I2,',',I2,')=(',1PE14.7,',',1PE14.7,')')
55    TYPE 59
59    FORMAT (1X)
      GO TO 10

C  TINY VALUES (DUE TO NUMERICAL ERROR) IN MATRIX SHOULD BE
C  SET TO 0 BY HAND.  ONCE THAT IS DONE, COME HERE.
C  SOLVE DIAGONAL 6 X 6 BOUNDARY VALUE MATRIX FOR RATIO
C  OF CONSTANTS C.
31    PAUSE 'DID YOU SET 0 IN MATRIX?'
      C(4,6)=CMPLX(0.,0.)
      C(5,6)=C(4,6)
      C(6,6)=C(4,6)
      DO 35 I=1,6
```

```
35      CTMP(I)=0.
        CTMP(6)=1.
        DO 38 I=5,1,-1
        DO 37 J=I+1,6
37      CTMP(I)=CTMP(I)-C(I,J)*CTMP(J)
38      CTMP(I)=CTMP(I)/C(I,I)
        DO 39 I=1,6
39      CFIN(IORD(I))=CTMP(I)
        DO 41 I=1,6
41      TYPE 42,I,CFIN(I)
42      FORMAT (' CFIN(',I1,')=('1PE14.7,',','1PE14.7,')')
99      STOP
        END

C  CG COMPUTES ENTRIES OF 6 X 6 MATRIX WHICH DESCRIBES
C  BOUNDARY CONDITIONS, AND THEN CALLS CDET TO FIND VALUE
C  OF DETERMINANT.
        FUNCTION CG(X)
        IMPLICIT COMPLEX(C)
        COMMON CLAM(3),CZ(10,10),CK(6)
        DIMENSION CA(20,20),CE(6),CC(3,6)
        CI=CMPLX(0.,1.)
        DO 110 J=1,6
        CE(J)=CEXP(CI*CK(J)*X)
110     CONTINUE
        DO 130 I=1,3
        DO 120 J=1,6
        CC(I,J)=CZ(I,(J+1)/2)
120     CONTINUE
130     CONTINUE
        DO 150 I=1,3
        DO 140 J=1,6
        CA(I,J)=CK(J)*CC(I,J)
        CA(I+3,J)=CK(J)*CC(I,J)*CE(J)
140     CONTINUE
150     CONTINUE
        CALL CDET(CA,6,CVAL)
        CG=CVAL
        RETURN
        END

        SUBROUTINE CDET(CS,NSIZE,CVAL)     !COMPLIMENTS OF R. J. HAYDEN.

C  CDET DIAGONALIZES (IN THE BEST WAY) THE MATRIX.
C  IT GIVES BACK VALUE OF DETERMINANT.
        IMPLICIT COMPLEX (C)
        COMMON /CD/ C(6,6),IORD(6)
        DIMENSION CS(20,20)
        DO 80 I=1,NSIZE
        DO 70 J=1,NSIZE
```

```
      C(I,J)=CS(I,J)
70    CONTINUE
80    CONTINUE
      DO 84 I=1,6
84    IORD(I)=I
      CSIGN=1.
      DO 140 K=1,NSIZE-1
      AMAX=CABS(C(K,K))
      JMAX=K
      DO 90 J=K+1,NSIZE
      ATRY=CABS(C(K,J))
      IF(ATRY.LE.AMAX) GO TO 90
      AMAX=ATRY
      JMAX=J
90    CONTINUE
      IF(AMAX.EQ.0.) GO TO 160
      IF(JMAX.EQ.K) GO TO 110
      CSIGN=-CSIGN
      DO 100 I=1,NSIZE
      CSAVE=C(I,K)
      C(I,K)=C(I,JMAX)
      C(I,JMAX)=CSAVE
100   CONTINUE
      ISAV=IORD(K)
      IORD(K)=IORD(JMAX)
      IORD(JMAX)=ISAV
110   DO 130 I=K+1,NSIZE
      CR=C(I,K)/C(K,K)
      C(I,K)=0.
      DO 120 J=K+1,NSIZE
      C(I,J)=C(I,J)-C(K,J)*CR
120   CONTINUE
130   CONTINUE
140   CONTINUE
      CVAL=CSIGN
      DO 150 K=1,NSIZE
142   IF (CABS(CVAL).LT.1.E20)  GO TO 144
      CVAL=CVAL/1.E10
      TYPE 148
148   FORMAT (' DET VALUE HAS BEEN DECREASED')
      GO TO 142
144   CVAL=CVAL*C(K,K)
150   CONTINUE
155   RETURN
160   CVAL=0.
      RETURN
      END
```

```
C  SMALLK.FOR

C  SMALLK NEEDS LIB:IMSL/LIB PROGRAMS TOO.
C  THIS PROG IS BASICALLY THE FIRST PART OF SMALL, WITH
C  THE FEATURE OF BEING ABLE TO CHANGE THE DIFFUSION
C  COEFFICIENTS AND THEN RUN FOR A RANGE OF F'S.
C  SMALLK SOLVES THE EIGENVALUE PROBLEM, GIVING EIGENVALUES
C  AND VECTORS.  EXISTENCE OF A POSITIVE, REAL EIGENVALUE
C  INDICATES THAT A FIRST ORDER SOLUTION EXISTS.
C  CRITICAL LENGTH IS PI/(SQRT(EIGENVALUE)).
      IMPLICIT DOUBLE PRECISION(A-B,D-H,O-Y)
      IMPLICIT COMPLEX(C)
      COMPLEX A(10,10),B(10,10),EIGA(10),EIGB(10),WK(10,20)
      DIMENSION LAB(13),CFIN(6),CTMP(6)
      COMMON CLAM(3),CZ(10,10),CK(6)
      COMMON /CD/ C(6,6),IORD(6)
      DATA LAB/'  S    Q    W   SREC AL   ET   RO   ADOT EDOT RDOT
      1 DFAL DFET DFRO'/

C  INITIALIZE
      REWIND 21                        !(FOR21.DAT SET BY PARSET)
      READ (21,410,ERR=460) AA,BB,HPL,VK1,VK2,VK3,VK4,VK5,F,H0,
      1  EPS,MF,DIFX,DIFY,DIFZ          !H0, EPS, MF ARE GEAR PARAMS
410   FORMAT(11D,I,3D)
      VK1=VK1*HPL*HPL
      VK2=VK2*HPL
      VK3=VK3*HPL
      Q=2.*VK1*VK4*AA/(VK2*VK3*BB)
      S=DSQRT(VK3*BB/(VK1*AA))
      SREC=1./S
      TCON=DSQRT(VK1*VK3*AA*BB)
      W=VK5/TCON
      TYPE 500,LAB(1),S
      TYPE 500,LAB(2),Q
      TYPE 500,LAB(3),W
      TYPE 500,LAB(4),SREC
300   TYPE 305
305   FORMAT (' ENTER DIFX,Y,Z')     !ENTER DIFFUSION COEFS.
      ACCEPT 310,DIFX,DIFY,DIFZ
310   FORMAT (3E)
      DIFAL=100.*DIFX/TCON
      DIFET=100.*DIFY/TCON
      DIFRO=100.*DIFZ/TCON
      TYPE 500,LAB(11),DIFAL
      TYPE 500,LAB(12),DIFET
      TYPE 500,LAB(13),DIFRO
      DO 600 G=.45,.55,.01    !!!!!!! SET F RANGE HERE  !!!!!
      F=G
      TYPE 615,F
```

```
615   FORMAT (/,' F= ',F6.3)
400   AL=((1.D0-F-Q)+DSQRT((Q+F-1.D0)**2+4.D0*(F+1.D0)*Q))/(2.D0*Q)
      ET=(F*AL/(1.+AL))
      RO=AL
C450  TYPE 500,LAB(5),AL
C     TYPE 500,LAB(6),ET
C     TYPE 500,LAB(7),RO
500   FORMAT(1X,A5,' = ',1PD)
      GO TO 480
460   PAUSE 'WHERE IS FILE'
      STOP

C  USE IMSL  ROUTINE EIGZC TO SOLVE THE EIGENVALUE MATRIX PROBLEM
C  AX = BX.  PRINTS EIGENVALUES AND EIGENVECTORS.

480   IJOB=2
      IA=10
      IB=10
      IZ=10
      NMAT=3
      DO 130 I=1,NMAT
      DO 120 J=1,NMAT
      B(I,J)=CMPLX(0.,0.)
120   CONTINUE
130   CONTINUE
      B(1,1)=DIFAL/S
      B(2,2)=DIFET*S
      B(3,3)=DIFRO/W
      A(1,1)=-ET-2.*Q*AL+1.
      A(1,2)=1.-AL
      A(1,3)=0.
      A(2,1)=-ET
      A(2,2)=-(AL+1.)
      A(2,3)=F
      A(3,1)=1.
      A(3,2)=0.
      A(3,3)=-1.
      CALL EIGZC(A,IA,B,IB,NMAT,IJOB,EIGA,EIGB,CZ,IZ,WK,INFER,IER)
      DO 220 J=1,NMAT
      CLAM(J)=EIGA(J)/EIGB(J)
      CK(2*J-1)=CSQRT(CLAM(J))
      CK(2*J)=-CK(2*J-1)
      RLAM=REAL(CLAM(J))
      AILAM=AIMAG(CLAM(J))
      TYPE 180,J,RLAM,AILAM
180   FORMAT(' LAMBDA(',I2,')=(',1PE14.7,',',1PE14.7,')')
220   CONTINUE
600   CONTINUE
      GO TO 300          !BR BACK IF ALL YOU WANT IS EIGENVALUES.
C                           IF YOU WANT MORE, KILL THIS BRANCH.
```

```
      DO 185 L=0,1
      K=2*J-1+L
      RLK=REAL(CK(K))
      AIK=AIMAG(CK(K))
185   TYPE 187,K,RLK,AIK
187   FORMAT('    K(',I2,')=(',1PE14.7,',',1PE14.7,')')
      TYPE 190,J
190   FORMAT(' EIGENFUNCTION #',I2)
      DO 210 I=1,NMAT
      RX=REAL(CZ(I,J))
      AIX=AIMAG(CZ(I,J))
      TYPE 200,I,RX,AIX
200   FORMAT(' X(',I2,')=(',1PE14.7,',',1PE14.7,')')
210   CONTINUE
C20   CONTINUE
      TYPE 230,INFER
230   FORMAT(' INFER=',I5)
      P=REAL(WK(1,1))
      TYPE 235,P
235   FORMAT(' PERFORMANCE INDEX=',1PE14.7)

      GO TO 300
      END
```

```
C  SSSOL.FOR

C  SSSOL IS HANDY FOR GETTING VALUES OF CONSTANTS FOR A SET
C  OF PARAMS.  IT WILL ALSO GIVE YOU THE TIME DERIVATIVES
C  FOR STEADY-STATE VALUES AND FOR VALUES NEAR THE STEADY-STATE.
      IMPLICIT DOUBLE PRECISION(A-H,O-Y)
      DIMENSION LAB(13)
      DATA LAB/'  S     Q     W    SREC AL    ET    RO    ADOT EDOT RDOT
      1 DFAL DFET DFRO'/
100   REWIND 21                        !(FOR21.DAT SET BY PARSET)
      READ (21,110,ERR=460) A,B,HPL,VK1,VK2,VK3,VK4,VK5,F,H0,
      1  EPS,MF,DIFX,DIFY,DIFZ          !H0, EPS, MF ARE GEAR PARAMS
110   FORMAT(11D,I,3D)
      VK1=VK1*HPL*HPL           !SET CONSTANTS FOR DIF. EQ.'S
      VK2=VK2*HPL
      VK3=VK3*HPL
      FF=F
      Q=2.*VK1*VK4*A/(VK2*VK3*B)
      S=DSQRT(VK3*B/(VK1*A))
      SREC=1./S
      CON=DSQRT(VK1*VK3*A*B)
      W=VK5/CON
      DIFAL=100.*DIFX/CON
      DIFET=100.*DIFY/CON
      DIFRO=100.*DIFZ/CON
      TYPE 500,LAB(1),S
      TYPE 500,LAB(2),Q
      TYPE 500,LAB(3),W
      TYPE 500,LAB(4),SREC
      TYPE 500,LAB(11),DIFAL
      TYPE 500,LAB(12),DIFET
      TYPE 500,LAB(13),DIFRO
400   AL=((1.D0-FF-Q)+DSQRT((Q+FF-1.D0)**2+4.D0*(FF+1.D0)*Q))/(2.D0*Q)
      ET=(FF*AL/(1.+AL))
      RO=AL
450   TYPE 500,LAB(5),AL
      TYPE 500,LAB(6),ET
      TYPE 500,LAB(7),RO
500   FORMAT(1X,A5,' = ',1PD)
430   FORMAT(1X'ALPHA= 'E10.4,'  ETA= 'E10.4,'  RHO= 'E10.4)
      ADOT=S*(ET+AL*(-ET+1.-Q*AL))
      EDOT=SREC*(-ET*(1.+AL) +FF*RO)
      RDOT=W*(AL-RO)
      TYPE 500,LAB(8),ADOT
      TYPE 500,LAB(9),EDOT
      TYPE 500,LAB(10),RDOT
      TYPE 550
550   FORMAT ('  ADD TO EACH')
      ACCEPT 580,AINC,EINC,RINC
```

```
580   FORMAT (3D)
      AL=AL+AINC
      ET=ET+EINC
      RO=RO+RINC
      GO TO 450
600   STOP
460   PAUSE 'WHERE IS FILE'
      GO TO 600
      END
```