APPENDIX A

Qualitative Analysis of Differential Equations

G. Bard Ermentrout and Joel E. Keizer

Nonlinear ordinary differential equations are notoriously difficult or impossible to solve analytically. On the other hand, the solution to linear equations like those encountered in the kinetic model of the GLUT transporter can be expressed in terms of simple functions, and their behavior analyzed using standard results from linear algebra. In the first two sections, we cover some basic ideas in linear algebra and a review of power series. These are concepts that are needed in the sections that follow. In Section A.3 we summarize the main results for linear equations with two dependent variables. Although one often encounters models like the GLUT transporter that involve more than two variables, the basic ideas for two-variable ODEs carry over more or less unchanged for larger sets of linear equations. Thus the intuition gained from understanding simple two variable ODEs is enormously useful in understanding more complicated models. To help to develop this intuition, we introduce the notion of the *phase plane* and use phase plane analysis to help understand the solution of two-variable linear equations in Section A.4. Another reason for focusing on linear equations is that the stability of nonlinear ODEs can be understood by examining the behavior of small deviations around steady or oscillatory states. In Section A.4.2 we show how the properties of linearized equations can be used to understand stability of steady states for a system of nonlinear equations such as a membrane with a gated ion channel.

Although we have tried to include a good brief synopsis of the most important tools used in this book, more study on these topics may be necessary to fully appreciate some of the more complex mathematical concepts. For more information on these topics, please see the suggested readings listed at the end of this appendix.

A.1 Matrix and Vector Manipulation

Matrices can be multiplied, added, multiplied by scalar numbers, and differentiated according to the rules of linear algebra. Here we summarize these results for the 2×2 matrices and two-component column vectors . The matrix $\hat{A}$ multiplying the vector $\mathbf{x}$ acts as a linear operator that produces a new vector, $\mathbf{z}$, according to the formula

$$\mathbf{z} = \hat{A}\mathbf{x} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} a_{11}x_1 + a_{12}x_2 \\ a_{21}x_1 + a_{22}x_2 \end{pmatrix}. \tag{A.1}$$

It can be verified using (A.1) that the identity matrix $\hat{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ leaves vectors unchanged, i.e., $\mathbf{z} = \hat{I}\mathbf{x} = \mathbf{x}$. Matrices can be added together, as can vectors, using the rules

$$\hat{A} + \hat{B} = \begin{pmatrix} a_{11} + b_{11} & a_{12} + b_{12} \\ a_{21} + b_{21} & a_{22} + b_{22} \end{pmatrix} \quad \text{and} \quad \mathbf{x} + \mathbf{y} = \begin{pmatrix} x_1 + y_1 \\ x_2 + y_2 \end{pmatrix}. \tag{A.2}$$

To multiply either a matrix or a vector by a scalar c, each component is multiplied by c, e.g.,

$$c\hat{A} = c \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = \begin{pmatrix} ca_{11} & ca_{12} \\ ca_{21} & ca_{22} \end{pmatrix}. \tag{A.3}$$

Differentiation of matrices and vectors is also carried out on each component separately. Thus

$$d\mathbf{x}/dt = \begin{pmatrix} dx_1/dt \\ dx_2/dt \end{pmatrix}. \tag{A.4}$$

The *trace*, *determinant*, and *discriminant* are important scalars that characterize matrices and that appear in the solution to (A.15). We use the shorthand notation $tr\hat{A}$ for the trace of $\hat{A}$, $det\hat{A}$ for its determinant, and $disc\hat{A}$ for the descriminant. In terms of matrix elements they are defined as

$$tr\hat{A} = a_{11} + a_{22} \tag{A.5}$$

$$det\hat{A} = a_{11}a_{22} - a_{21}a_{12} \tag{A.6}$$

$$disc\hat{A} = (tr\hat{A})^2 - 4det\hat{A} \tag{A.7}$$

For example, for the matrix

$$\hat{A} = \begin{pmatrix} 1 & -1 \\ 3 & 6 \end{pmatrix}, \tag{A.8}$$

$tr\hat{A} = 7$, $\det\hat{A} = 9$, and $disc\hat{A} = 13$.

The inverse of a matrix is the generalization of division by a number. The inverse of $\hat{A}$ is written as $\hat{A}^{-1}$ and is a matrix with the property that

$$\hat{A}\hat{A}^{-1} = \hat{A}^{-1}\hat{A} = \hat{I} \tag{A.9}$$

with $\hat{I}$ the identity matrix. The inverse of a matrix is useful in solving linear algebraic equations. For example, the solution of the linear equation

$$\hat{A}\mathbf{x} = \mathbf{y} \tag{A.10}$$

is

$$\mathbf{x} = \hat{A}^{-1}\mathbf{y}, \tag{A.11}$$

which can be verified by multiplying both sides of (A.10) on the left by $\hat{A}^{-1}$ and using (A.9). For a 2×2 matrix it is easy to verify by carrying out the matrix multiplication in (A.9) that if $\hat{A}$ is not *singular*, i.e., as long as $det\hat{A} \neq 0$, then

$$\hat{A}^{-1} = \frac{1}{det\hat{A}} \begin{pmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{pmatrix}. \tag{A.12}$$

A.2 A Brief Review of Power Series

One of the most useful tehniques in applied mathematics is the method of power series expansion. The basic idea is that many functions can be expressed as a series in one or more variables. For example, the familiar exponential function can be written as

$$e^t = 1 + \frac{t}{1!} + \frac{t^2}{2!} + \cdots + \frac{t^n}{n!} + \cdots,$$

or more compactly as

$$e^t = \sum_{n=0}^{\infty} \frac{t^n}{n!},$$

where we define $0! = 1$. The series converges for all t both real and complex. Given a function $f(t)$ and a point $t = t_0$, suppose that all the derivatives of f at the point t_0 are defined. Then we can formally develop a power series approximation of the function f around the point t_0. The formal power series is

$$f(t) = \sum_{n=0}^{\infty} f^{(n)}(t_0) \frac{(t - t_0)^n}{n!}. \tag{A.13}$$

Here $f^{(k)}(t_0)$ is the kth derivative of the function f evaluated at the point t_0. That is, given the derivatives of a function at a point, we can approximate the function over some interval containing that point by using a series approximation. This series is called a

Taylor series of f about the point t_0. When the point t_0 is 0 the series is often called a Maclaurin series.

If all of the derivatives of the given function exist at the point t_0, then the finite approximation to the Taylor series,

$$S_N(t) = \sum_{n=0}^{N} f^{(n)}(t_0)\frac{(t-t_0)^n}{n!},$$

is also defined for all t, since it is just a finite sum of polynomials. We say that the series *converges* for t in some interval I containing t_0 if the limit of $S_N(t)$ exists as $N \to \infty$ for all t in I. The interval I for which convergence is obtained is called the *interval of convergence* for the series. For the exponential series given above, the interval of convergence is the whole real line. Infinite series do not always converge on the whole line. For example, the geometric series

$$S(t) = 1 + t + t^2 + t^3 + \cdots + t^n + \cdots$$

converges for $|t| < 1$. A useful test for the convergence of a series of the form

$$S = \sum_{n=0}^{\infty} a_n$$

is the ratio test. Let $R_n = |a_{n+1}/a_n|$. If

$$\lim_{n\to\infty} R_n < 1,$$

then the series converges. Let us apply this to the exponential series above. Since $a_n = t^n/n!$, we have $|a_{n+1}/a_n| = t/(n+1)$, and the limit of this as n goes to infinity is zero for any finite t, so that the series converges for all t.

Here are some examples. Let us find a series approximation for $f(0) = \sin(t)$ about $t = 0$. Note that $f(0) = 0, f'(0) = 1, f''(0) = 0, f'''(0) = -1$, and the higher derivatives just cycle among these numbers. That is, derivatives of order 1, 5, 9, etc., are equal to 1, those of order 3, 7, 11, etc., are equal to -1, and all others are zero. Thus

$$\sin(t) = t - \frac{t^3}{3!} + \frac{t^5}{5!} + \cdots + (-1)^m \frac{t^{2m+1}}{(2m+1)!} + \cdots.$$

The ratio $|a_{m+1}/a_m| = t^2/(2m+2)(2m+3)$, which tends to 0 as m goes to infinity, so the sine series converges for all t.

You can similarly verify that

$$\cos(t) = \sum_{m=0}^{\infty} (-1)^m \frac{t^{2m}}{(2m)!}.$$

As a final example, consider the series for the square root function evaluated at $t = 1$. We have the following first few derivatives:

$$f(1) = 1, \quad f'(1) = \frac{1}{2}, \quad f''(1) = \frac{-1}{2}\frac{1}{2}, \quad f'''(1) = \frac{-3}{2}\frac{-1}{2}\frac{1}{2}.$$

Thus the nth derivative ($n > 1$) is

$$c_n = (-1)^{n+1} \frac{(2n-3)(2n-5)\cdots 1}{2^n}.$$

Thus,

$$\sqrt{(t)} = 1 + \frac{1}{2}(t-1) + \sum_{n=2}^{\infty} (-1)^{n+1} \frac{c_n (t-1)^n}{n!}.$$

We can apply the ratio test to this, noting that $|c_{n+1}/c_n| = n - \frac{1}{2}$, so that

$$R_n = |t-1| \left(n - \frac{1}{2} \right) / (n+1).$$

As $n \to \infty$ this ratio goes to $|t-1|$. The interval of convergence satisfies $|t-1| < 1$, or $0 < t < 2$.

A.3 Linear ODEs

The simplest time-dependent differential equations to solve are linear in the dependent variables and of first order in the time. *First order* implies that only the first time derivative appears on the left–hand side of the equations, and linear implies that the right–hand side is a linear function of the dependent variables. The most general equations of this type in n variables have the form

$$\begin{aligned} dx_1/dt &= a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n + y_1, \\ dx_2/dt &= a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n + y_2, \\ &\vdots \\ dx_n/dt &= a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n + y_n. \end{aligned} \tag{A.14}$$

Here we consider only the case where the a_{ij} and y_i are parameters that are independent of time. For simplicity, we focus in this chapter on the special case of two variables, which shares the main features of the more general case. Using the column vector and matrix notation introduced in Section A.1, we can write these equations concisely as

$$d\mathbf{x}/dt = \hat{A}\mathbf{x} + \mathbf{y} \tag{A.15}$$

with

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \quad \hat{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}. \tag{A.16}$$

Using the rules for differentiation of vectors, matrix multiplication, and vector addition it is easy to verify that the vector equation (A.15), when written in terms of component vectors, is the special case of (A.14) for two variables, i.e.,

$$dx_1/dt = a_{11}x_1 + a_{12}x_2 + y_1, \tag{A.17}$$

$$dx_2/dt = a_{21}x_1 + a_{22}x_2 + y_2. \tag{A.18}$$

In this book we will be interested in equations for which there is a *steady–state* solution, $\mathbf{x}^{ss}$. In particular, we will be interested in what happens near these steady–state solutions. This is a solution that is independent of time, so that setting the left–hand side of (A.15) equal to zero and rearranging gives $\mathbf{y} = -\hat{A}\mathbf{x}^{ss}$. Using this expression we can eliminate $\mathbf{y}$ from (A.15) by defining $\mathbf{x}' = \mathbf{x} - \mathbf{x}^{ss}$, to get

$$d\mathbf{x}'/dt = \hat{A}\mathbf{x}'. \tag{A.19}$$

This has the same form as (A.15) with $\mathbf{y} = \mathbf{0}$, but now this equation concerns deviations, or perturbations, from the steady–state solution. In the next section we will show how to solve equations of this type using simple algebra.

A.3.1 Solution of Systems of Linear ODEs

The simplest way to solve an equation like (A.15) is to use the component form of the equation to obtain a new equation that is of second order in time. For simplicity we focus on the special case that $\mathbf{y} = \mathbf{0}$ and introduce the notation $\dot{x}_1$ for the first time derivative and $\ddot{x}_1$ for the second, so that

$$\dot{x}_1 = a_{11}x_1 + a_{12}x_2, \tag{A.20}$$

$$\dot{x}_2 = a_{21}x_1 + a_{22}x_2. \tag{A.21}$$

Thus differentiating both sides of (A.20) with respect to time gives

$$\begin{aligned}
\ddot{x}_1 &= a_{11}\dot{x}_1 + a_{12}\dot{x}_2 \\
&= a_{11}\dot{x}_1 + a_{12}(a_{21}x_1 + a_{22}x_2) \\
&= a_{11}\dot{x}_1 + a_{12}a_{21}x_1 + a_{22}(a_{12}x_2) \\
&= a_{11}\dot{x}_1 + a_{12}a_{21}x_1 + a_{22}(\dot{x}_1 - a_{11}x_1) \\
&= a_{11}\dot{x}_1 + a_{22}\dot{x}_1 - a_{11}a_{22}x_1 + a_{12}a_{21}x_1,
\end{aligned} \tag{A.22}$$

where in rewriting the right–hand side we have first used (A.21) to replace $\dot{x}_2$ and then used (A.20) to eliminate the term $a_{12}x_2$. Using the last equality in (A.22) and the definitions of the trace and determinant of $\hat{A}$ in Section A.1 gives a second–order equation for x_1:

$$\ddot{x}_1 - \left(tr\hat{A}\right)\dot{x}_1 + \left(det\hat{A}\right)x_1 = 0. \tag{A.23}$$

Using similar manipulations, an identical second order equation can be derived for x_2:

$$\ddot{x}_2 - \left(tr\hat{A}\right)\dot{x}_2 + \left(det\hat{A}\right)x_2 = 0. \tag{A.24}$$

To solve (A.23), we try the exponential function $x_1(t) = c\exp(\lambda t)$ ($c \neq 0$). Substituting this into the left–hand side of (A.23) gives

$$c\lambda^2\exp(\lambda t) - c\lambda\exp(\lambda t)tr\hat{A} + c\exp(\lambda t)det\hat{A}. \tag{A.25}$$

Therefore, $c\exp(\lambda t)$ is a solution to (A.23) if

$$\lambda^2 - (tr\hat{A})\lambda + det\hat{A} = 0. \tag{A.26}$$

This is called the *characteristic equation* of the matrix $\hat{A}$. It is a quadratic equation in λ with the well-known solution

$$\lambda_{\pm} = \frac{tr\hat{A} \pm \left(\left(tr\hat{A}\right)^2 - 4det\hat{A}\right)^{1/2}}{2}. \tag{A.27}$$

For example, if $\hat{A} = \begin{pmatrix} 1 & -1 \\ 3 & 6 \end{pmatrix}$, then $tr\hat{A} = 7$, $\det\hat{A} = 9$, and $\lambda_{\pm} = \left(7 \pm \sqrt{13}\right)/2$. As long as $\lambda_+ \neq \lambda_-$, then the solution to the characteristic equation gives two independent solutions to (A.23). In this case, because (A.23) is linear, it is easy to verify that the sum of these two solutions, $c_+\exp(\lambda_+ t) + c_-\exp(\lambda_- t)$, is also a solution.

The argument of the square root in (A.27) is the *discriminant* of the matrix $\hat{A}$ defined in Section A.1. As long as $disc\hat{A} \neq 0$, then it is clear from (A.27) that there are two independent solutions for $x_1(t)$. Using the result in (A.23), it follows that the solution for $x_2(t)$ also has the same form. We write this concisely as

$$x_i(t) = b_{i1}\exp(\lambda_+ t) \; + b_{i2}\exp(\lambda_- t) \tag{A.28}$$

with $i = 1, 2$. The values of the constants b_{1i} and b_{2i} need to be chosen to satisfy the initial conditions. This is easily worked out, for example, for x_1. Recall that there are two initial conditions, $x_1(0)$ and $x_2(0)$, since there are two equations. Using (A.20) and (A.28) it follows that

$$x_1(0) = b_{11} + b_{12}, \tag{A.29}$$

$$\dot{x}_1(0) = a_{11}x_1(0) + a_{12}x_2(0) = b_{11}\lambda_+ + b_{12}\lambda_-. \tag{A.30}$$

Since $x_1(0), x_2(0), \lambda_+$, and λ_- are known, (A.29) and the second equality in (A.30) provide two independent equations for the two unknowns b_{11} and b_{12}. Solving these using elementary algebra gives

$$b_{11} = \frac{\dot{x}_1(0) - \lambda_- x_1(0)}{\lambda_+ - \lambda_-}, \tag{A.31}$$

$$b_{12} = \frac{-\dot{x}_1(0) + \lambda_+ x_1(0)}{\lambda_+ - \lambda_-}. \tag{A.32}$$

Since $\lambda_+ \neq \lambda_-$, the denominators of these equations are different from zero.

The time dependence of $x_1(t)$ is strongly dependent on the nature of the characteristic values. There are three possibilities that are determined by the sign of the discriminant and the trace. If $disc\hat{A} > 0$, then according to (A.27) the two characteristic values will be distinct real numbers, since for a matrix with real components, $tr\hat{A}$ is a real number. However, if $disc\hat{A} < 0$, then the roots will be conjugate complex numbers (if $tr\hat{A} \neq 0$), and will in fact be pure imaginary numbers (if $tr\hat{A} = 0$).

When the characteristic values are complex, (A.28) can be expressed in terms of sines, cosines, and exponentials. This follows from the representation of the exponential of a complex number $r + i\omega$ (with $i = \sqrt{-1}$) as

$$\exp(r + i\omega) = \exp(r)\exp(i\omega) = \exp(r)\left(\cos(\omega) + i\sin(\omega)\right). \tag{A.33}$$

If we express the characteristic values in this fashion as $\lambda_{\pm} = r \pm i\omega$, then it is not difficult to show using (A.28)–(A.32) that

$$x_1(t) = \exp(rt)\left(x_1(0)\cos(\omega t) - \frac{(\dot{x_1}(0) - rx_1(0))}{\omega}\sin(\omega t)\right). \tag{A.34}$$

Straightforward differentiation of this expression verifies that it satisfies the initial conditions and that it is identical to the expression in (A.28).

The solution to (A.23) is slightly different when $disc\hat{A} = 0$. In this case, according to (A.27) $\lambda_+ = \lambda_-$, and there is only a single characteristic value $\lambda = tr\hat{A}/2$. In this case, in addition to $c\exp(\lambda t)$ there is a second solution to (A.23), which is $c't\exp(\lambda t)$. This can be verified using the facts that $tr\hat{A} = 2\lambda$ and (since $disc\hat{A} = (tr\hat{A})^2 - 4det\hat{A} = 0$) that $det\hat{A} = \lambda^2$. Substituting these expressions for $tr\hat{A}$ and $det\hat{A}$ into (A.23) gives

$$\ddot{x}_1 - 2\lambda\dot{x}_1 + \lambda^2 x_1 = 0. \tag{A.35}$$

It is easy to show then by substitution that $c't\exp(\lambda t)$ solves (A.35). Thus when $disc\hat{A} = 0$, the general solution to (A.23) is

$$x_1(t) = b_{11}\exp(\lambda t) + b_{12}t\exp(\lambda t). \tag{A.36}$$

Using the initial conditions

$$x_1(0) = b_{11}, \tag{A.37}$$

$$\dot{x}_1(0) = a_{11}x_1(0) + a_{12}x_2(0) = \lambda b_{11} + b_{12}, \tag{A.38}$$

it is easy to show that in this case

$$b_{11} = x_1(0), \tag{A.39}$$

$$b_{12} = \frac{a_{11} - a_{22}}{2}x_1(0) + a_{12}x_2(0). \tag{A.40}$$

Although matrices with a vanishing discriminant are not typical, it is easy to construct specific examples, e.g., $\hat{A} = \begin{pmatrix} -3 & 5 \\ 0 & -3 \end{pmatrix}$. For this matrix $tr\hat{A} = -6$, $det\hat{A} = 9$, $disc\hat{A} = 0$, and $\lambda = -3$, and the solution for x_1 is easily found from (A.36)–(A.40) to be

$$x_1(t) = (x_1(0) + 5tx_2(0))\exp(-3t). \tag{A.41}$$

A.3.2 Numerical Solutions of ODEs

Although we have characterized the solutions to (A.20)–(A.21) analytically, it is just as easy to solve them numerically by creating a system of ODEs in a numerical solver.

Consider the matrix $\hat{A} = \begin{pmatrix} 1 & -1 \\ 3 & 6 \end{pmatrix}$, with characteristic values $\lambda_{\pm} = (7 \pm \sqrt{13})/2 =$ 5.31 and 1.70. Since these values are distinct, we know that the solution is a sum of two exponentials with positive exponents. Thus as long as $x_1(0) \neq 0$ *and* $x_2(0) \neq 0$, the magnitude of x_1 and x_2 will increase exponentially with time. This is shown in Figure A.1A, where the solution generated using a numerical package is plotted. Notice that x_1 rapidly declines, whereas x_2 increases even more rapidly. The difference is due to the coefficients of the two exponentials, which can be calculated explicitly from the formulae in (A.31) and (A.32).

Figure A.1B shows another solution for the matrix $\hat{A} = \begin{pmatrix} -1 & -1 \\ 5 & 1 \end{pmatrix}$ for which $disc\hat{A} = -16$ and therefore $\lambda_{\pm} = \pm 4i$. Since the characteristic values are imaginary, (A.34) shows that the solution is a sum of sines and cosines, as shown in the figure. The matrix $\hat{A} = \begin{pmatrix} -2 & -1 \\ 4 & 1 \end{pmatrix}$, on the other hand, has complex solutions with a real part equal to $-\frac{1}{2}$. Thus the solution for this matrix will be a sum of sines and cosines multiplied by $\exp(-t)$. Numerical solution of the equations leads to the damped oscillations shown in Figure A.1C.

A.3.3 Eigenvalues and Eigenvectors

The characteristic values of $\hat{A}$ are also the *eigenvalues* corresponding to the *eigenvectors* of the matrix. An eigenvector e_λ of $\hat{A}$ has the property that

$$\hat{A}e_\lambda = \lambda e_\lambda, \tag{A.42}$$

where λ is a number called the eigenvalue. In other words, the matrix $\hat{A}$ transforms an eigenvector into a constant multiple of the eigenvector. This equation can be rewritten in component form as

$$\begin{aligned} (a_{11} - \lambda)e_{1\lambda} + a_{12}e_{2\lambda} &= 0, \\ a_{21}e_{1\lambda} + (a_{22} - \lambda)e_{2\lambda} &= 0. \end{aligned} \tag{A.43}$$

The only way to have a nonzero solution to this equation for $\mathbf{e}_\lambda$ is that the determinant of the coefficients on the left-hand side of (A.43) vanishes, i.e.,

$$det\hat{A} = \begin{pmatrix} a_{11} - \lambda & a_{12} \\ a_{21} & a_{22} - \lambda \end{pmatrix} = 0. \tag{A.44}$$

Expanding the determinant, one obtains the characteristic equation (A.26), which shows that the eigenvalues are the same as the characteristic values of the matrix.

The eigenvectors of a matrix are defined only up to a multiplicative constant, since if $\mathbf{e}_\lambda$ satisfies (A.42), then so does $c\mathbf{e}_\lambda$. As long as $\lambda_+ \neq \lambda_-$, then it is not difficult to verify

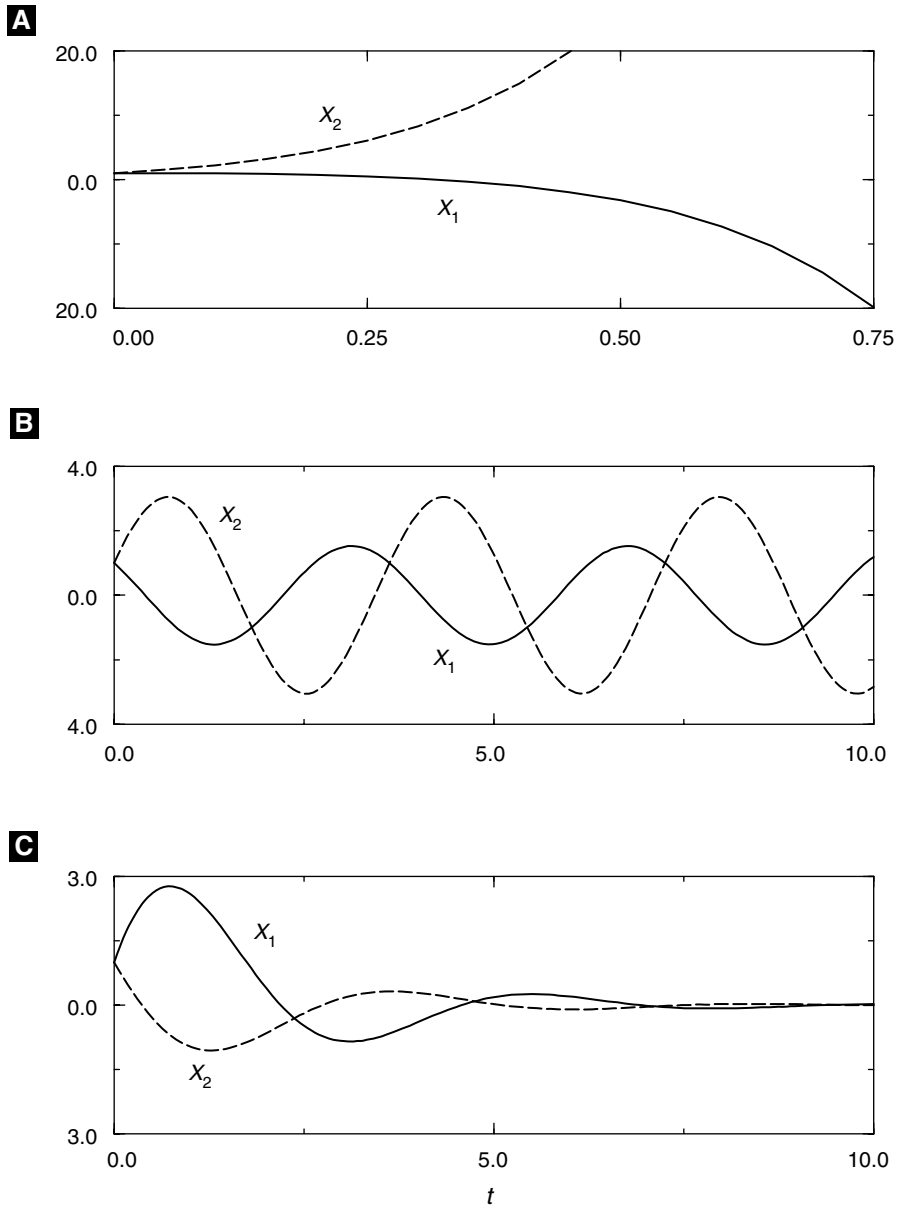

Figure A.1 Solution of 2×2 linear equations using a numerical solver. Panels (A)–(C) give the time course of the solutions for the three matrices described in the text.

that the eigenvectors are given by the simple formula

$$\mathbf{e}_\lambda = \begin{pmatrix} 1 \\ (\lambda - a_{11})/a_{12} \end{pmatrix}. \tag{A.45}$$

For example, for the matrix $\hat{A} = \begin{pmatrix} 1 & 1 \\ 2 & 1 \end{pmatrix}$ $tr\hat{A} = 2$, $det\hat{A} = -1$, and $\lambda_{\pm} = 1 \pm \sqrt{2}$. Applying (A.45), the eigenvectors are

$$\mathbf{e}_{+} = \begin{pmatrix} 1 \\ \sqrt{2} \end{pmatrix} \text{ and } \mathbf{e}_{-} = \begin{pmatrix} 1 \\ -\sqrt{2} \end{pmatrix}. \tag{A.46}$$

A useful property of eigenvectors of $\hat{A}$ is that if $\mathbf{e}_{\lambda}$ is the initial condition for (A.20) and (A.21), then the solution is

$$\mathbf{x}(t) = \exp(\lambda t)\mathbf{e}_{\lambda}. \tag{A.47}$$

This result can be verified by differentiating the right–hand side of (A.47) to get $\dot{\mathbf{x}}(t) = \lambda \exp(\lambda t)\mathbf{e}_{\lambda}$ and then noticing that (A.42) implies that

$$\hat{A}\mathbf{x}(t) = \hat{A}\exp(\lambda t)\mathbf{e}_{\lambda} = \exp(\lambda t)\hat{A}\mathbf{e}_{\lambda} = \lambda \exp(\lambda t)\mathbf{e}_{\lambda}, \tag{A.48}$$

which shows that $\mathbf{x}(t)$ solves the equations. We apply this result in the following sections.

A.4 Phase Plane Analysis

Obtaining a "solution" to first–order ODEs means that you have expressed all of the dependent variables as functions of time. In the case of the 2×2 linear equations in Section A.3, this means that we have the time series for x_1 and x_2. A great deal can be learned about these solutions by plotting the dependent variables as a function of time as done in Figure A.1. However, there are other ways of plotting solutions that give additional insight. For example, one can plot $\dot{x}_1$ versus time, or some function of x_1 and x_2 versus time. Perhaps the most useful plot is a *phase plane* plot, in which x_2 is plotted versus x_1 with time serving only as a parameter, as shown in Figure A.2 for the numerical solutions shown in Figure A.1. This type of plot represents the *trajectory* of the solution, just as the arc of a baseball thrown in the air is a trajectory in three dimensional space.

Technically, the phase plane (or phase space for more than two variables) is a Cartesian plane with coordinates (x_1, x_2). Since the initial condition for the ODEs is arbitrary, any one of these points could be the initial point of a trajectory like those in Figure A.2. Continuing the analogy of phase space trajectories to the trajectory of a baseball, it makes sense to associate a *velocity* with the trajectory that goes through a point in phase space. This can be done directly using the differential equations, since the right–hand sides of the equations are explicit expressions for $\dot{x}_1$ and $\dot{x}_2$ as functions of x_1 and x_2. Thus for the matrix $\hat{A} = \begin{pmatrix} -2 & -1 \\ 4 & 1 \end{pmatrix}$ that gives rise to the trajectory in Figure A.2C, the x_1 component of the velocity at the point (x_1, x_2) is $-2x_1 - x_2$, whereas the x_2 component of the velocity is $4x_1 + x_2$. For the initial point $(0.5, 0.5)$ of the

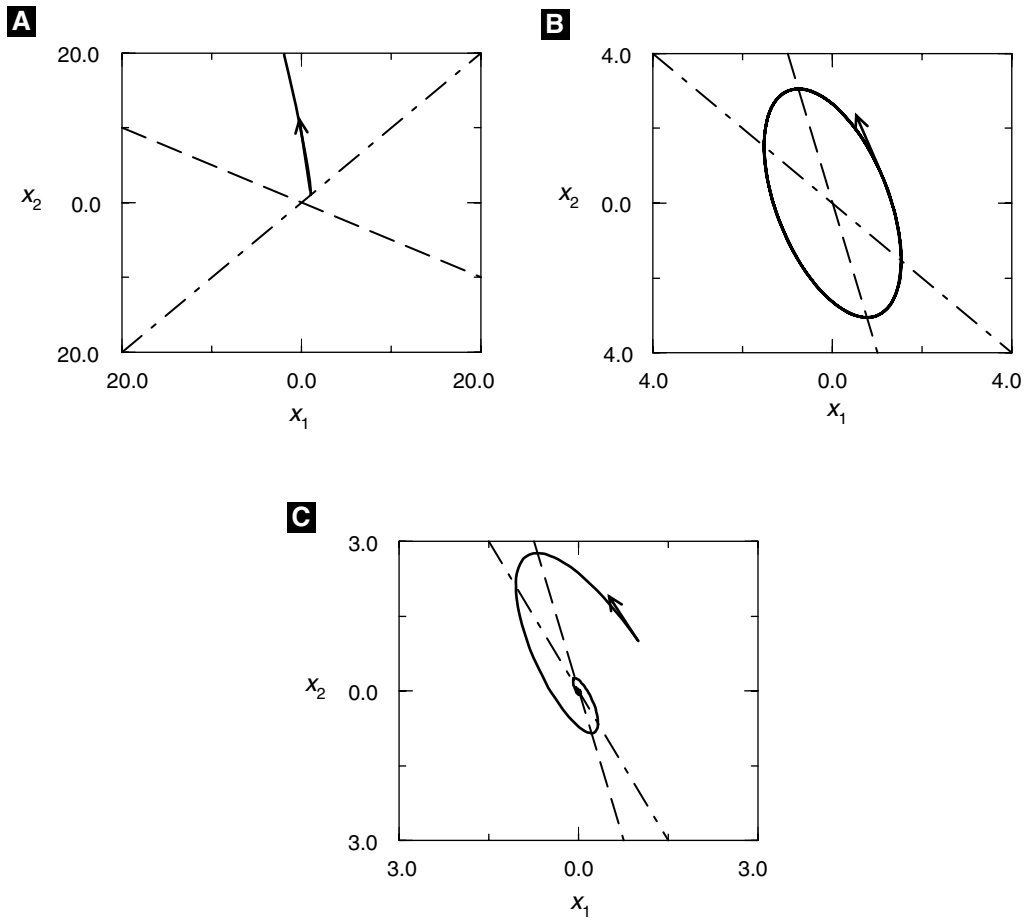

Figure A.2 The three solutions in Figure A.1(A)–(C) represented here in phase plane plots in corresponding (A)–(C). The arrow represents the direction of the initial point on the trajectory, which is given by the full line. The dashed line is the x_2 nullcline, the broken dashed line is the x_1 nullcline, and their intersection is the steady state, which is unstable in (A), marginally stable in (B), and stable in (C).

trajectory in Figure A.2C the velocity vector at that point has components $(-1.5, 2.5)$. In the figure, the head of the arrow on the velocity vector indicates its direction, and the length is proportional to its magnitude. Just as the velocity of a baseball is parallel to its trajectory, so is the velocity vector in phase space parallel to its trajectory.

There are a number of important curves and points in the phase plane that are defined by the differential equations. *Isoclines* are lines in the phase plane where the time rate of change of a variable is constant. For example, for the matrix in the previous paragraph, the isoclines for x_1 are defined by $c = -2x_1 - x_2$, i.e., $x_2 = -2x_1 + c$, and the isoclines for x_1 are given by $x_2 = -4x_1 + c$, where c is a constant. A particularly

useful isocline is the *nullcline*, for which the time rate of change is zero, i.e., $c = 0$. So for this special case the nullclines are given by the straight lines through the origin, $x_2 = -2x_1$ and $x_2 = -4x_1$, shown in Figure A.2C. It is straightforward to show that the nullclines for the general 2×2 linear equations (A.17)–(A.18) are also straight lines. Since $\dot{x}_1 = 0$ on the x_1 nullcline, x_1 cannot decrease if the trajectory crosses the nullcline from the right and cannot increase if the trajectory crosses it from the left. This means, as can be verified by looking at Figure A.2, that the trajectory must cross the x_1 nullcline perpendicular to the x_1 axis. Similary, the trajectory crosses the x_2 nullcline perpendicular to the x_2 axis.

Steady states are defined as points in the phase space at which both $\dot{x}_1 = 0$ and $\dot{x}_2 = 0$. These points, which are also known as *singular points*, *equilibrium points*, or *stationary points*, have the property that neither variable changes as a function of time. They are determined graphically by the intersection of the nullclines. However, just because the variables do not change in time at a steady state does not mean that trajectories starting from nearby points will end up at the steady state. Three different situations are illustrated in Figure A.2. In panel A the steady state is at the origin, $(0, 0)$. However, the trajectory starting at $(0.5, 0.5)$ grows without bound. In panels B and C the steady states are also at the origin, but the trajectory in B circles the origin periodically, whereas in C it spirals into the steady state.

A.4.1 Stability of Linear Steady States

As we saw in the preceding section, a steady state may or may not be an *attractor* for nearby trajectories; i.e., just because an initial condition is close to the steady state, it does not mean that after a time the trajectory will approach the steady state. However, when this is the case, the steady state is said to be *stable and attractive* or *asymptotically stable*. Three qualitatively different behaviors near steady state are illustrated by the solutions of the linear ODEs in Figure A.2. The matrix for the ODEs in panel A has positive eigenvalues, and the trajectory is repelled, not attracted, by the steady state. So the steady state in panel A is asymptotically unstable. In panel B the trajectory is circular and periodically returns to the initial condition $(0.5, 0.5)$. In this case the steady state is neither attractive nor repulsive and is said to be *neutrally stable*. Note that if the steady state of a nonlinear problem is determined to be neutrally stable by finding the eigenvalues of the linearized problem, we are not able to conclude *anything* about the stability of the steady state. Neutral stability is a borderline case, and the nonlinear parts of the equations can affect the stability in either direction. In these cases, more analysis must be done to determine the stability. Finally, in panel C the trajectory spirals into the steady state, which is stable and attractive.

The attentive reader may have noticed a correlation between the eigenvalues of the three matrices represented in Figure A.2 and the stability of the steady states. Indeed, unstable states of linear equations are characterized by at least one eigenvalue with a positive real part. If, in addition, both eigenvalues are positive, as in Figure A.2A, then the state is called an *unstable node*. An asymptotically stable state like that in Figure

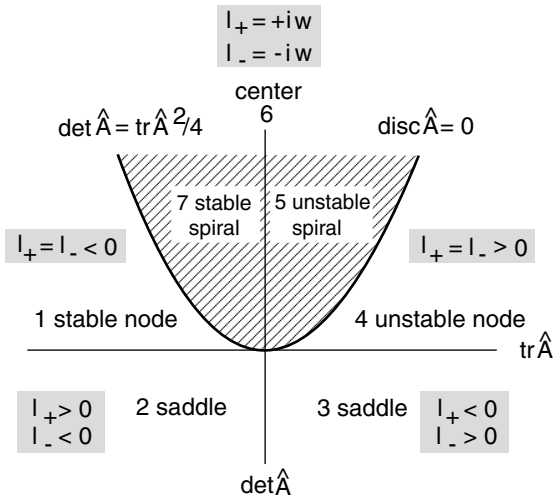

Figure A.3 Graphical representation of the stability properties of 2×2 matrices. The trace is plotted on the x-axis and the determinant on the y-axis. The eigenvalues in the cross–hatched region are complex, and real elsewhere. These two regions are separated by the parabola $det\hat{A} = tr\hat{A}^2/4$ on which $disc\hat{A} = 0$. Seven regions with various degrees of stability are indicated by the sign of the eigenvalues.

A.2C, on the other hand, has negative real parts for all of its eigenvalues. Marginal (or neutral) stability occurs when the real part of a pair of eigenvalues vanishes, as is the case in Figure A.2B. A two–variable linear equation has only two eigenvalues, and a marginally stable steady state implies sinusoidal solutions. Another name for a marginal state for a two–variable system is a *center*.

Because we have at our disposal the analytical form of the solutions for 2×2 linear equations, it is possible to give a complete description of the stability of their steady states. Figure A.3 gives a graphical representation of the stability behavior of a matrix $\hat{A}$ as a function of the trace (plotted on the x-axis) and the determinant (plotted on the y-axis). The $tr\hat{A}$, $det\hat{A}$plane in Figure A.3 is divided into seven distinct regions separated by the two axes and the parabola $det\hat{A} = tr\hat{A}^2/4$, which is the curve on which $disc\hat{A} = 0$. According to the expression for the characteristic values in (A.27), $\lambda_+ = \lambda_-$ on the parabola, and in the quadrant with $tr\hat{A} > 0$ the eigenvalues are both positive, whereas for $tr\hat{A} < 0$ both eigenvalues are negative. Marginal stability occurs when the real part of both eigenvalues is zero, i.e., when $tr\hat{A} = 0$ and $det\hat{A} > 0$, which occurs on the positive y-axis. Using (A.27) it is easy to verify that complex eigenvalues occur only in the cross-hatched region above the parabola (since $disc\hat{A} < 0$ there). In that region to the right of the y-axis, $tr\hat{A} > 0$, the eigenvalues have positive real parts, and the steady states are unstable spirals (region 5), whereas in region 7 the spirals are stable. When $tr\hat{A}$ and $det\hat{A}$ have values in regions 1 and 4, the steady state is a stable or unstable node, respectively. Below the x-axis (where $det\hat{A} < 0$) the steady states are unstable with the property that they have two real eigenvalues, one positive and one negative. Unstable states like this are called *saddle* points, because trajectories that start in the direction of the positive eigenvector recede from the steady state exponentially. Trajectories along the direction of the negative eigenvector move toward the steady state, also exponentially.

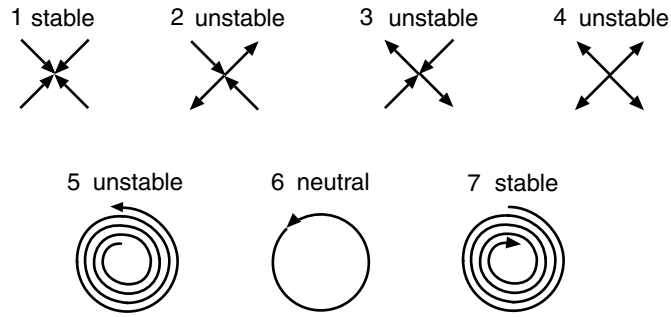

Figure A.4 Schematic representation of phase space trajectories near the steady states in the seven regions shown in Figure A.3. Unstable states have trajectories that diverge from the steady state, whereas stable steady states have converging trajectories, and neutrally stable states are surrounded by closed trajectories. The states shown in 2 and 3 are saddle points, with both converging and diverging trajectories in the directions of the eigenvectors of the matrix.

Using Figure A.3 we can classify the qualitative behavior of phase plane trajectories for 2×2 linear ODEs based on the value of their trace and determinant. Representative trajectories are illustrated in Figure A.4 for each of the seven regions in Figure A.3. Region 1 is a stable node, and the two trajectories correspond to the directions of the two stable eigenvectors, which have velocity vectors directed at the steady state. Regions 2 and 3 are saddle points, with eigenvectors that move toward or away from the steady state, whereas the unstable node in region 4 has both eigenvectors moving away. Regions 5 and 7 have trajectories that spiral away from or toward the steady state. In region 6 the trajectories are circles, corresponding to sinusoidal oscillations.

A.4.2 Stability of a Nonlinear Steady States

What we have learned about stability of steady states for linear systems can be transferred partially to nonlinear ODEs. To be specific, let us consider a biological membrane with a gated ion channel. To do this we combine a model of ion gating with an expression that governs the membrane potential (see Chapter 1 and Chapter 2). For simplicity, we will consider only one conductance. If n represents the gating variable and V the voltage, then the two are coupled by the differential equations

$$CdV/dt = -gn(V - V_{\text{rev}}) + I_{\text{app}}, \tag{A.49}$$

$$dn/dt = -(n - n_\infty)/\tau, \tag{A.50}$$

where V_{rev} is the reversal potential. We assume that n_∞ has the following voltage dependence:

$$n_\infty = \frac{1}{1 + \exp(-(V + V_{0.5})/S)} \tag{A.51}$$

with $V_{0.5}$ and S positive constants. Equations (A.49) and (A.50) are both nonlinear due to the factor $n(V - V_{\text{rev}})$ in (A.49) and the voltage dependence of n_∞ in (A.50).

To analyze the stability of the steady states of these equations we first must find the steady states by setting the right–hand sides of the equations equal to zero. This gives

$$g n^{\text{ss}}(V^{\text{ss}} - V_{\text{rev}}) = I_{\text{app}} \tag{A.52}$$

$$n^{\text{ss}} = n_\infty(V^{\text{ss}}), \tag{A.53}$$

which can be written as a single nonlinear equation to solve for V^{ss}:

$$\frac{I_{\text{app}}}{g} = \frac{V^{\text{ss}} - V_{\text{rev}}}{1 + \exp(-(V^{\text{ss}} + V_{0.5})/S)}. \tag{A.54}$$

This equation cannot be solved in closed form, and a much simpler way to locate the steady state is graphically in the (V, n) phase plane using the nullclines. Setting the left–hand sides of (A.49) and (A.50) separately equal to zero and solving for n as a function of V gives

$$n = \frac{I_{\text{app}}}{g(V - V_{\text{rev}})} \quad (V\text{-nullcline}), \tag{A.55}$$

$$n = n_\infty = \frac{1}{1 + \exp(-(V + V_{0.5})/S)} \quad (n\text{-nullcline}). \tag{A.56}$$

The V- and n-nullclines are plotted in Figure A.5A, along with representative trajectories. Due to the nonlinearities in (A.49) and (A.50) the nullclines are curved rather than straight lines. This curvature influences the shape of the trajectories, which must

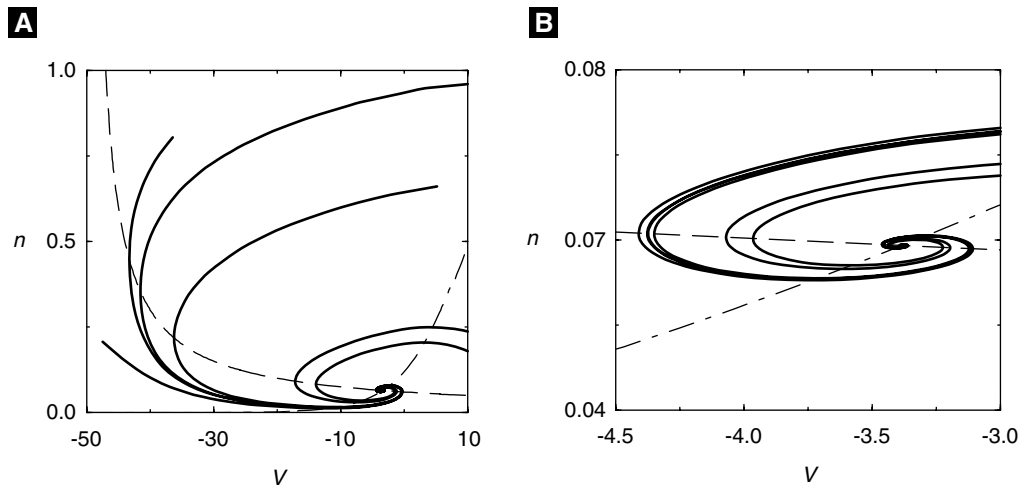

Figure A.5 Phase plane plots for (A.49)-(A.51) showing typical trajectories (full lines), the V-nullcline (dashed line), and the n-nullcline (broken dashed line). (B) is zoomed-in around the steady state, illustrating that the nullclines are approximately straight lines near the steady state.

cross the nullcline perpendicular to the axis of the variable. Close to the steady state, however, both nullclines become approximately straight lines, as is seen in Figure A.5B, which is the same phase plane as in Figure A.5A, but zoomed in around the steady state.

If we restrict the initial conditions for trajectories to be close to the steady state, then the nonlinear equations are well approximated by a 2×2 linear system. This can be seen in detail if we define as new variables $x_1 = V - V^{ss}$ and $x_2 = n - n^{ss}$, the deviations of the voltage and gating variable from their steady–state values. Since the steady–state values are constants, it follows that $dx_1/dt = dV/dt$ and $dx_2/dt = dn/dt$, so that we can use (A.49) and (A.50) to obtain differential equations for x_1 and x_2. In particular, if the initial conditions are close to the steady state, then we can substitute $V = V^{ss} + x_1$ and $n = n^{ss} + x_2$ into the right–hand sides of (A.49) and (A.50) and then use a Taylor series expansion in the small deviations x_1 and x_2. Explicitly:

$$\begin{aligned} dx_1/dt &= \left(g(n^{ss} + x_2)(V^{ss} + x_1 - V_{rev}) + I_{app}\right)/C \\ &= [gn^{ss}(V^{ss} - V_{rev}) + I_{app}]/C + (gn^{ss}x_1 + g(V^{ss} - V_{rev})x_2)/C \\ &\quad + gx_1x_2/C, \qquad \text{(A.57)} \\ dx_2/dt &= -\left(n^{ss} + x_2 - n_\infty(V^{ss} + x_1)\right)/\tau \\ &= -[n^{ss} - n_\infty(V^{ss})]/\tau + (dn_\infty/dV)^{ss}x_1/\tau - x_2/\tau \\ &\quad + \text{higher–order terms in } x_1. \qquad \text{(A.58)} \end{aligned}$$

In the second equality in both (A.57) and (A.58) the terms in square brackets vanish because of the steady–state conditions in (A.52) and (A.52); the second terms are linear in x_1 and x_2; and the third terms are quadratic or of higher order in x_1 and x_2. Thus keeping the lowest–order terms gives the linear equations

$$dx_1/dt = (gn^{ss}/C)x_1 + (g(V^{ss} - V_{rev})/C)x_2, \qquad \text{(A.59)}$$

$$dx_2/dt = (dn_\infty/dV)^{ss}x_1/\tau - x_2/\tau. \qquad \text{(A.60)}$$

Once the elements of the matrix of this 2×2 linear equation have been evaluated, the behavior of the solution in a neighborhood of the steady state can be evaluated. This type of linear analysis, which gives information only about trajectories nearby the steady state, is called *linear stability analysis*.

The trajectories in Figure A.5B make it clear that the steady state is asymptotically stable, and according to the catalogue of possibilities in Figure A.4, the steady state is a stable spiral. It is also possible to find the steady states numerically and, in addition, determine the stability of the steady state by numerical evaluation of the eigenvalues. Combining the analytical tools developed in this chapter with the numerical tools available in various software packages, we are ready to explore the dynamics of a variety of cellular and neural dynamical systems in the remaining chapters.

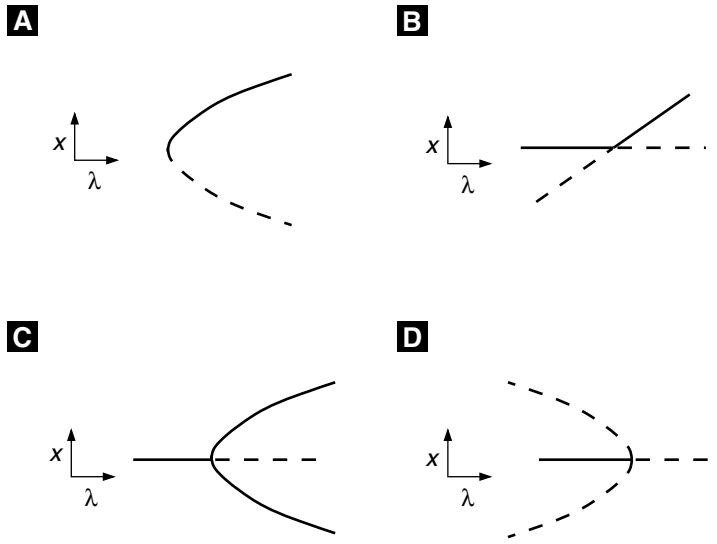

Figure A.6 Bifurcations of new fixed points: (A) saddle–node bifurcation, (B) transcritcial bifurcation, (C) supercritical pitchfork bifurcation, (D) subcritical pitchfork bifurcation. Stable fixed points are solid, and unstable are dashed.

A.5 Bifurcation Theory

In many systems of differential equations there are parameters that we would like to vary. As these parameters vary, we want to know whether the solutions to the equations remain similar in nature. For example, as current is injected into a cell, we want to know if the cell will remain at rest or whether some other phenomena that are qualitatively different will take place. The changes in the qualitative nature of solutions to differential equations as a parameter varies is called bifurcation theory. In this section we will review simple bifurcations from equilibrium of ordinary differential equations. Bifurcation from equilibrium solutions is intimately related to the stability of equilibria, a subject described earlier in this chapter. Suppose that we have found an equilibrium solution to a system of differential equations and study its stability as some relevant parameter varies. The stability is determined from the eigenvalues of the linearized system. There are two simple ways that stability can change as a parameter varies: (i) A real negative eigenvalue can cross through zero and become positive; (ii) a pair of complex conjugate eigenvalues with negative real parts crosses through the imaginary axis and becomes a pair of complex eigenvalues with positive real parts. In a fully nonlinear system these changes in stability will often lead to the appearance of new solutions to the differential equations. Because these are new branches of solutions that were not there previously, the system has undergone a qualitative change in behavior.

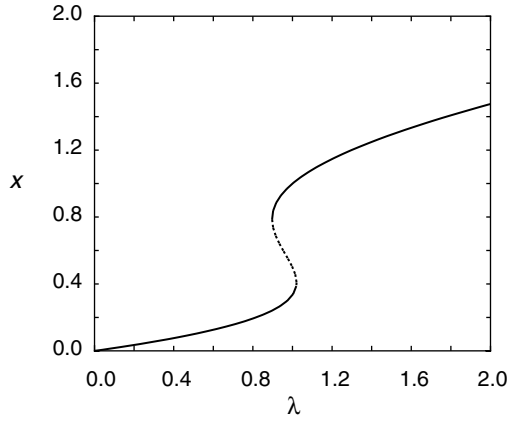

Figure A.7 Numerically computed bifurcation diagram for the autocatalytic chemical model. Stable fixed points are solid, and unstable are dashed.

A.5.1 Bifurcation at a Zero Eigenvalue

Consider the differential equation

$$\frac{dx}{dt} = \lambda - x^2, \tag{A.61}$$

where λ is a parameter. For $\lambda < 0$ there are no real equilibria. However, if $\lambda > 0$, then there are two equililibrium solutions, $x = \pm\sqrt{\lambda}$. Consider the case $\lambda > 0$. The linearization about the positive fixed point is $-2\sqrt{\lambda}$. Thus, it is a stable fixed point. Note that as λ tends to zero the eigenvalue of this 1×1 matrix goes to zero. Any time an eigenvalue of the linearization around an equilibrium point crosses zero, we can expect to see more than one fixed point in the neighborhood of the parameter. The graph of the equilibrium solution against the parameter along with the stability information is called a *bifurcation diagram*. Figure A.6A shows the bifurcation diagram for (A.61). This type of bifurcation is called a saddle node. The autocatalytic chemical model

$$\frac{dx}{dt} = \lambda - 6x + \frac{10x^2}{1+x^2}$$

has two saddle-node bifurcations as the input λ increases from 0. For $0 < \lambda < 0.9$ there is a single equilibrium point. At $\lambda \approx 0.9$ a new pair of equilibria appear at $x \approx 0.8$. As λ continues to increase these new equilibria drift apart, and at $\lambda \approx 1.02$ the leftmost equilibrium merges with the middle one and disappears at $x \approx 0.4$. We can use a numerical package to draw a complete bifurcation diagram of this. Figure A.7 illustrates the complete bifurcation diagram. Note the two saddle-node bifurcations; for λ between these two values there are three equilibria (two stable and one unstable), while for λ outside the two values there is a unique stable equilibrium point. Techniques from nonlinear analysis can be used to show that every saddle-node bifurcation (no matter what the dimension of the system) is equivalent and can be transformed into (A.61).

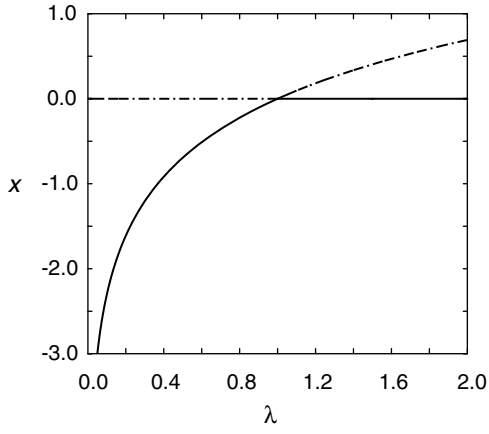

Figure A.8 Numerically computed bifurcation diagram for the example transcritical bifurcation. Stable fixed points are solid, and unstable are dashed.

Consider next the differential equation

$$\frac{dx}{dt} = \lambda x - x^2. \tag{A.62}$$

In some model systems there is always a "trivial" equilibrium point, no matter what the parameter is. (In this case, 0 is always a solution.) For $\lambda < 0$, $x = 0$ is a stable equilibrium, and for $\lambda > 0$ it is unstable. The equilbrium point $x = \lambda$ is unstable (stable) for $\lambda < 0$ ($\lambda > 0$). Thus as λ crosses zero the two fixed points "exchange stability." This is called a *transcritical* or *exchange of stability* bifurcation. Figure A.6B illustrates this bifurcation. For example, consider the system

$$\frac{dx}{dt} = x(1 - \lambda y), \quad \frac{dy}{dt} = e^{-x} - y.$$

Clearly, one fixed point is $(0, 1)$, and the Jacobian matrix for the linearization about this point is

$$J = \begin{pmatrix} 1-\lambda & 0 \\ -1 & -1 \end{pmatrix}.$$

The eigenvalues are -1 and $-1 + \lambda$. Thus at $\lambda = 1$ we expect that there could be a bifurcation. It is not a saddle node since there always exists the trivial equilibrium $(0, 1)$. Since there are no additional symmetries in the problem (see below), it is likely a transcritical bifurcation. The diagram is shown in Figure A.8. As with the saddle-node bifurcation, all transcritical bifurcations can be transformed into (A.62) near the bifurcation.

Many biological and chemical systems are characterized by symmetries. In this case, the behavior as parameters vary is analogous to

$$\frac{dx}{dt} = x(\lambda \pm x^2). \tag{A.63}$$

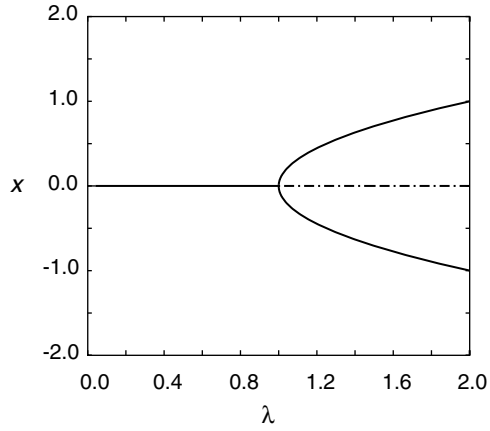

Figure A.9 Numerically computed bifurcation diagram for the coupled system showing a pitchfork bifurcation. Stable fixed points are solid, and unstable are dashed.

As with the transcritical bifurcation, $x = 0$ is always a solution to this problem. For $\lambda < 0$, the fixed point 0 is stable, and for $\lambda > 0$ this trivial fixed point loses stability. At $\lambda = 0$ the linearized system has a zero eigenvalue. There can be *two* additional solutions depending on λ. Unlike the two bifurcations we previously described, the sign of the nonlinearity is important in this one. If we take the negative sign in (A.63), then the diagram in Figure A.6C is obtained. The new solutions are $x = \pm\sqrt{\lambda}$; they are both stable, and they occur for $\lambda > 0$. The branches open in the same direction as the trivial fixed point loses stability. This bifurcation is called a *supercritical pitchfork bifurcation*. If we take instead the positive sign for the nonlinearity in (A.63), then there are two solutions $x = \pm\sqrt{-\lambda}$, and they occur for $\lambda < 0$. As can easily be shown, they are both unstable. This is called a *subcritical pitchfork bifurcation*, since the branches open in the direction opposite from the change of stability of the trivial equilibrium point.

For example, consider the simple coupled system

$$\frac{dx}{dt} = -x + \lambda \frac{y}{1+y^2}, \quad \frac{dy}{dt} = -y + \lambda \frac{x}{1+x^2}.$$

It is easy to see that $x = y = 0$ is always a fixed point and that it is stable as long as $\lambda < 1$. At $\lambda = 1$ the Jacobian matrix has a zero eigenvalue, so we expect a bifurcation to occur. Figure A.9 shows that it is a supercritical pitchfork bifurcation. Every system that has a pitchfork bifurcation can be transformed into (A.63) near the bifurcation point. Unlike the saddle-node and the transcritical bifurcations, the details of the nonlinearity are crucial for determing the stability of the new branches of solutions.

A.5.2 Bifurcation at a Pair of Imaginary Eigenvalues

Limit cycles and periodic solutions are extremely important in physiology. Thus, one is often interested in whether or not they occur in a given system. Unlike fixed points that can be found exactly or graphically, it is much more difficult to determine whether or not there are limit cycles in a system. There is one method that is arguably the best and

perhaps only systematic method of finding parameters where there may be periodic solutions in any system of differential equations. The existence of periodic solutiuons emanating from a fixed point is established from the Hopf bifurcation theorem, which we now state.

Hopf bifurcation theorem. *Suppose that* $X' = F(X, \lambda)$ *has an isolated fixed point* $X_0(\lambda)$. *Let* $A(\lambda)$ *be the linearized matrix about this fixed point. Suppose that the matrix A has a pair of complex conjugate eigenvalues* $\alpha(\lambda) \pm i\omega(\lambda)$. *Suppose the following conditions hold for some* λ_0*:*

1. $\alpha(\lambda_0) = 0$;
2. $\omega(\lambda_0) = \omega_0 > 0$;
3. $\nu \equiv d\alpha(\lambda)/d\lambda|_{\lambda=\lambda_0} \neq 0$;
4. $A(\lambda_0)$ *has no other eigenvalues with zero real part.*

Then, the system contains an isolated limit cycle for $|\lambda - \lambda_0|$ *small for either* $\lambda > \lambda_0$ *or for* $\lambda < \lambda_0$. *The magnitude of the limit cycle is proportional to* $\sqrt{|\lambda - \lambda_0|}$, *and the frequency is close to* ω_0. *If* $\nu > 0$ *and the limit cycle exists for* $\lambda > \lambda_0$ *or if* $\nu < 0$ *and the limit cycle exists for* $\lambda < \lambda_0$, *then it is stable. Otherwise, it is unstable.*

Thus, the best way to try to find periodic solutions in a system of differential equations is to look for parameter values where the stability of an equilibrium is lost as a complex conjugate pair of eigenvalues crosses the imaginary axis. For a two-

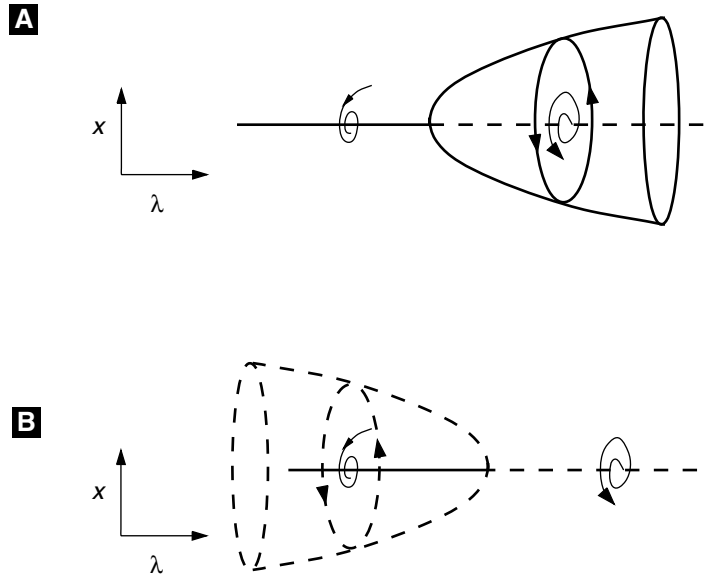

Figure A.10 Illustration of the Hopf bifurcation of limit cycles. As the parameter changes, a branch of periodic solutions emerges from the fixed point. (A) Supercritical emergence of stable limit cycles. (B) Subcritical emergence of unstable periodic orbits.

dimensional system, this situation occurs when the determinant of A is positive and the trace of A changes from negative to positive. The following system illustrates the theorem:

$$\begin{aligned} \frac{dx}{dt} &= \lambda x - y \pm x\left(x^2 + y^2\right), \\ \frac{dy}{dt} &= \lambda y + x \pm y\left(x^2 + y^2\right). \end{aligned} \tag{A.64}$$

Clearly, $(0,0)$ is always a fixed point. The eigenvalues of the linearization are $\lambda \pm i$, so that as λ goes from negative to positive, there is a pair of eigenvalues with imaginary real part at $\lambda = 0$. If we convert (A.65) to polar coordinates, $x = r\cos\theta$, $y = r\sin\theta$ then we obtain

$$\frac{dr}{dt} = r\left(\lambda \pm r^2\right), \quad \frac{d\theta}{dt} = 1.$$

The equation for r is just like (A.63), and thus the direction of bifurcation depends on the sign of the nonlinearity. We see that $r = \sqrt{\mp\lambda}$. Clearly, the solution to the θ equation is $\theta = t + C$, where C is an arbitrary constant. We conclude that if the nonlinearity has a positive sign, then there is an unstable periodic solution for $\lambda < 0$ given by $(x(t), y(t)) = \sqrt{-\lambda}(\cos(t + C), \sin(t + C))$. If the nonlinearity has a negative sign, then the limit cycle exists for $\lambda > 0$, and it is stable. Figure A.10 illustrates the behavior for both cases. We remark that every system that undergoes a Hopf bifurcation can be transformed to (A.65).

As an example, we consider the Brusselator, a classic model for chemical oscillations:

$$\frac{dx}{dt} = a - (b+1)x + x^2 y, \quad \frac{dy}{dt} = bx - x^2 y.$$

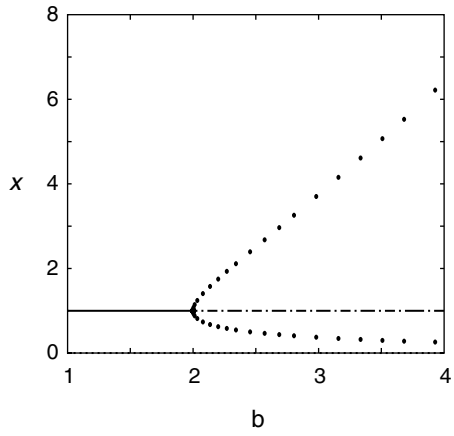

Figure A.11 Numerically computed bifurcation diagram for the Brusselator as the parameter b varies. Stable fixed points are solid, and unstable are dashed. Stable periodic orbits are filled circles.

The fixed points for this system are $(x, y) = (a, b/a)$, and the linearization about the fixed point is

$$A = \begin{pmatrix} b-1 & a^2 \\ -b & -a^2 \end{pmatrix}.$$

The determinant of A is $a^2 > 0$. The trace is $b - 1 - a^2$. Thus, if b is the parameter, then as b increases past $1 + a^2$ there will be a Hopf bifurcation. The full bifurcation diagram is shown in Figure A.11.

A.6 Perturbation Theory

As we have noted, nonlinear differential equations are not readily solved. In fact, even linear equations cannot always be solved in closed form if the coefficients are nonconstant in time. For this reason, one of the most powerful tools in applied mathematics is perturbation theory. In perturbation theory we look for very good approximate solutions. If some parameter in the equation is small, then a good initial approximation is to set it to zero. This can result in a simpler system of equations, which may be able to be solved. The idea is to assume that when the parameter is not zero, then we can use the simple case as a starting solution and expand the full solution in a power series in the small parameter. Typically, we need to expand the series to only one or two terms to see the dominant characteristics of the solution.

A.6.1 Regular Perturbation

Let us first consider the general solution and then work some examples. Consider

$$\frac{dx}{dt} = f(x, \epsilon), \tag{A.65}$$

where ϵ is a small parameter. Suppose that we can solve the equation with $\epsilon = 0$; that is, we can find a solution $x_0(t)$ to

$$\frac{dx}{dt} = f(x, 0).$$

Formally, let us look for a solution of the form

$$x(t, \epsilon) = x_0(t) + \epsilon x_1(t) + \epsilon^2 x_2(t) + \cdots \tag{A.66}$$

and substitute this into (A.65). This leads to a sequence of equations,

$$\frac{dx_0}{dt} = f(x_0, 0),$$
$$\frac{dx_1}{dt} = D_x f(x_0, 0) x_1 + D_\epsilon f(x_0, 0),$$

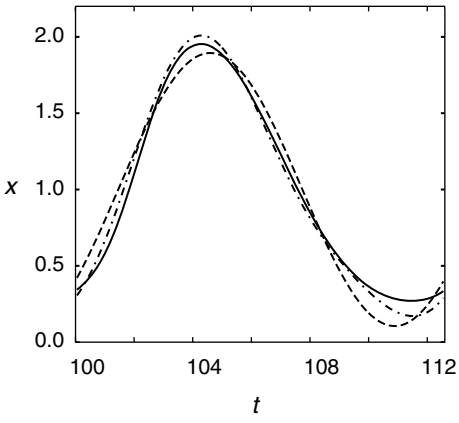

Figure A.12 The true solution (solid lines) and the first two terms in the perturbation series for the linear time-dependent logistic equation.

$$\frac{dx_2}{dt} = D_x f(x_0, 0)x_2 + D_{x\epsilon} f(x_0, 0)x_1 + \frac{1}{2}(D_{xx} f(x_0, 0)x_1^2 + D_{\epsilon\epsilon} f(x_0, 0),$$

where D_a is the derivative of f with respect to a evaluated at $x = x_0, \epsilon = 0$. Note that all but the first equation are linear. If the linear equation

$$\frac{dx}{dt} - D_x f(x_0, 0)x = g$$

is invertible, then we can continue this series method forever. As we will see later, when the equation is not invertible, then we run into problems, and other techniques are required. Another situation that can arise is that in which the small parameter multiplies dx_k/dt for one of the variables x_k. We will also consider this type of perturbation below.

Let us look at a simple example. Consider the differential equation for population growth subject to periodic forcing:

$$\frac{dx}{dt} = x(1 + \epsilon \sin(\omega t) - x).$$

We are interested in the steady–state behavior; thus we want to find solutions that are periodic or constant. Obviously, $x = 0$ is a solution for any ϵ, but this solution is of no interest, since it is unstable. When $\epsilon = 0$, another solution is $x = 1$. We will perturb from this solution:

$$x(t, \epsilon) = 1 + \epsilon x_1 + \epsilon^2 x_2 + \cdots.$$

Substituting this into the equation, we get

$$\frac{dx_1}{dt} = -x_1 + \sin(\omega t),$$
$$\frac{dx_2}{dt} = -x_2 + -x_1^2 + x_1 \sin(\omega t),$$

and so on. The x_1 equation has a periodic solution:

$$x_1(t) = \frac{\sin(\omega t) - \omega\cos(\omega t)}{1+\omega^2}.$$

Thus, to order ϵ,

$$x(t) = 1 + \epsilon\frac{\sin(\omega t) - \omega\cos(\omega t)}{1+\omega^2}.$$

To do even better, we can go to the next order. A simple bit of calculus shows that

$$x_2(t) = \frac{2\omega^4\cos(2\omega t) - 5\omega^3\sin(2\omega t) - 4\omega^2\cos(2\omega t) + \omega\sin(2\omega t)}{2 + 12\omega^2 + 18\omega^4 + 8\omega^6}.$$

Figure A.12 shows the numerical solution to the sample problem as well as the approximations $y_1(t) = 1 + \epsilon x_1(t)$ and $y_2(t) = y_1(t) + \epsilon^2 x_2(t)$ for $\epsilon = 1$ and $\omega = 0.5$. (For smaller values of ϵ and larger values of ω the approximation is much better.)

A.6.2 Resonances

In many applied problems the general perturbation scheme described above breaks down. Typically, this arises when there is a family of solutions to the lowest–order perturbation and the linear equations that arise from higher–order perturbations are not invertible.

A typical example of this would be perturbation of eigenvalues of a matrix. For example, suppose that the matrix A_0 is simple and we can find the eigenvalues easily. We now ask what the eigenvalues of the matrix $B = A_0 + \epsilon A_1$ are. Suppose that λ_0 is an eigenvalue and v_0 is the corresponding eigenvector. That is,

$$A_0 v_0 = \lambda_0 v_0.$$

To find the eigenvalue of B near λ_0 we suppose that both the eigenvalue and the eigenvector depend on ϵ:

$$v(\epsilon) = v_0 + \epsilon v_1 + \cdots,$$
$$\lambda(\epsilon) = \lambda_0 + \epsilon\lambda_1 + \cdots.$$

Making the substitutions, we get

$$(A_0 - \lambda_0 I)v_1 = \lambda_1 v_0 - A_1 v_0 \equiv w. \tag{A.67}$$

There are two unknowns, v_1 and λ_1. However, the matrix $C = A_0 - \lambda_0 I$ is not invertible, so we cannot expect to solve this unless λ_1 is chosen so that w is in the range of the matrix C. This condition uniquely determines the parameter λ_1. Then we can solve for v_1.

How do we know when a vector w is in the range of a matrix M? The following theorem tells us precisely the conditions:

Fredholm Alternative Theorem. *The matrix equation*

$$My = w$$

has a solution y if and only if $w \cdot q = 0$ *for every solution q to the equation* $M^* q = 0$. *The matrix* M^* *is the transpose complex conjugate of the matrix M.*

An analogous theorem holds for many other linear operators. Returning to (A.67), let q_0 be the solution to

$$C^T q_0 = 0, \qquad q_0 \cdot v_0 = 1.$$

Then the Fredholm alternative theorem implies that we must have

$$q_0 \cdot (\lambda_1 v_0 - A_1 v_0) = 0,$$

or

$$\lambda_1 = q_0 \cdot A_1 v_0.$$

Another classic example is to find a periodic solution to a weakly nonlinear differential equation. The van der Pol oscillator is the standard example:

$$\ddot{x} + x = \epsilon \dot{x}(1 - x^2). \tag{A.68}$$

We seek periodic solutions to this problem. Expanding $x(t)$ in ϵ,

$$x(t) = x_0(t) + \epsilon x_1(t) + \cdots,$$

and substituting into (A.68) we get

$$\begin{aligned} \ddot{x_0} + x_0 &= 0, \\ \ddot{x_1} + x_1 &= \dot{x_0}(1 - x_0^2). \end{aligned}$$

The solution to the first equation is

$$x_0(t) = A \cos t + B \sin t.$$

Note that we can rewrite this as $x_0(t) = C \cos(t + \phi)$, where ϕ is a phase shift. Since the equation is autonomous, there is always an arbitrary phase shift, so we can set this to zero. In other words, we can assume $x_0(t) = A \cos t$, where A is an arbitrary amplitude as yet unknown. The second equation is

$$\ddot{x_1} + x_1 = -A \sin t(1 - A^2 \cos t).$$

This does not generally have a periodic solution. In fact, it is easy to solve explicitly (using a symbolic algebra program, like Maple). The key point is that the solution will be of the form

$$x_1(t) = P(t) + tQ(t),$$

where $P(t), Q(t)$ are periodic. Unless $Q(t) = 0$, the perturbed solutions $x_1(t)$ will not be periodic, so we must make $Q(t) = 0$. A simple calculation reveals that

$$Q(t) = A\frac{4 - A^2}{8}\cos t.$$

Thus, we choose $A = 2$, and to lowest order

$$x(t) = 2\cos t.$$

A.6.3 Singular Perturbation Theory

In many physiological systems there are vast differences in the time scales involved in the phenomena. For example, in a bursting neuron there is the period between bursts compared with the interspike interval of the action potential within a burst. Some variables may act much more slowly than other variables, while others act much more rapidly. Consider, for example, the simple linear differential equation

$$\epsilon\frac{dx}{dt} = y - x, \quad \frac{dy}{dt} = -x,$$

along with initial conditions $y = 1, x = 0$. We can easily solve this exactly using the methods of the previous section for any value of ϵ. However, typically, in a real problem, the solutions are not so readily obtained. Let us suppose that we can set $\epsilon = 0$. Then we must have $0 = y - x$, or $x = y$. Thus our problem is now

$$\frac{dy}{dt} = -y, \quad y(0) = 1,$$

which has a solution $y(t) = \exp(-t)$. Furthermore, since $x = y$, we also have $x(t) = \exp(-t)$. Unfortunately, our "solution" does not satisfy the initial conditions $x(0) = 0$. Because we have reduced the order of the differential equation from 2 to 1, we cannot generally expect to find a solution for all initial conditions. This is why the problem is said to be *singular*.

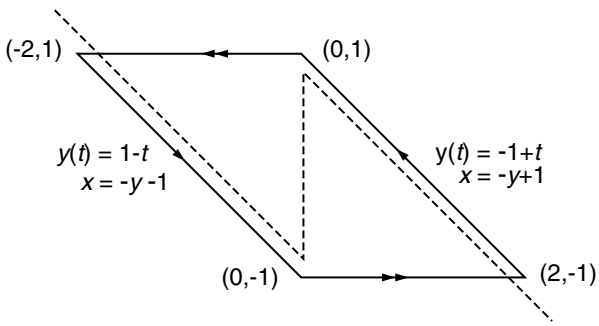

Figure A.13 x-nullclines for the relaxation oscillator example.

The way that we can fix this is to use a technique called matching. A complete description of matching goes well beyond this book, so we will just sketch this and another example. There are more examples throughout the text. The idea is to rescale time. Since the problems we are having occur at $t = t_0 = 0$, we introduce a new variable $\tau = (t - t_0)/\epsilon$. Under this change of variables our equation is

$$\frac{dX}{d\tau} = Y - X, \quad \frac{dY}{d\tau} = -\epsilon X.$$

(I have used capital letters to distinguish these solutions from the t-dependent solutions.) Now, we see that Y is "slow" in the new time scale. Set $\epsilon = 0$. This means that $dY/d\tau = 0$, so that Y is constant. The obvious constant to use is the initial value of Y, so we substitute $Y = 1$ into the X equation:

$$\frac{dX}{d\tau} = 1 - X, \quad X(0) = 0.$$

The solution to this is $X(\tau) = 1 - \exp(-\tau)$. Thus we have two sets of solutions, $(x(t), y(t))$ and $(X(\tau), Y(\tau))$. The (X, Y) solutions are valid for times near zero, and the (x, y) are valid for larger times. Thus, to obtain the full solution, we add these two together and subtract the "common" part. To see what the common part is, we replace τ by t/ϵ in the (X, Y) system and t by $\epsilon\tau$ in the (x, y) system. We take the limit as $\epsilon \to 0$ and obtain $(1, 1)$ for both sets of limits. This is the common part. Thus, our approximate solution is $(X(\tau) + x(t) - 1, Y(\tau) + y(t) - 1)$. Putting everything in terms of the original time, t, we obtain

$$x_c(t) = e^{-t} - e^{-t/\epsilon}, \quad y_c(t) = e^{-t}.$$

I close this section with another example that produces a singular nonlinear oscillator. The equations are

$$\epsilon\frac{dx}{dt} = -x + \text{sgn}(x) - y, \quad \frac{dy}{dt} = y + x,$$

where $\text{sgn}(x)$ is the signum function; it is $+1$ for $x > 0$ and -1 for $x < 0$. The nullclines are depicted in Figure A.13. For ϵ small, we expect that the solution will hug the x-nullcline, since we must have $-x + \text{sgn}(x) - y$ nearly zero. Setting $\epsilon = 0$ we must solve

$$-x + \text{sgn}(x) - y$$

for x in terms of y. Unfortunately, for y between -1 and 1 there are two roots $x = -y \pm 1$. For the moment, let us pick $x = -y + 1$. We must have $-y + 1 > 0$, since our choice of $+1$ for $\text{sgn}(x)$ assumes that $x > 0$. Substituting this into the y equation yields

$$\frac{dy}{dt} = 1,$$

so that $y(t) = y(0) + t$. Notice that as long as $y(t) < 1$, this is a valid solution, since $x > 0$. However, eventually $y(t)$ will exceed 1, and our root $x = -y + 1$ is no longer valid. So, what happens? Let t_0 be the time at which $y(t) = 1$. To see what happens, we must once

again introduce a scaled time $\tau = (t - t_0)/\epsilon$. Then our equations are

$$\frac{dX}{d\tau} = -X + \text{sgn}(X) - Y, \quad \frac{dY}{d\tau} = \epsilon(Y + X).$$

Setting $\epsilon = 0$, this means that Y must be constant. Since $y(t_0) = 1$, we will take $Y = 1$ as the constant. We must solve

$$\frac{dX}{d\tau} = -X + \text{sgn}(X) - Y, \quad X(0) = 0.$$

Note that for any $\tau > 0$, $X(\tau)$ is negative, so that $\text{sgn}(X) = -1$ and

$$X(\tau) = -2(1 - \exp(-\tau)).$$

This says that in the expanded time scale, $X(\tau)$ will drop from 0 down to -2. All the while, Y is essentially constant at 1. Once X has made the jump from 0 to -2, we can set $y = 1, x = -2$ and solve the $y(t)$ equation again. In this case, $x + y = -1$, since $\text{sgn}(x) = -1$, and we must solve

$$\frac{dy}{dt} = -1, \quad y(0) = 1.$$

The solution to this is $y(t) = 1 - t$. As above, this is valid only as long as $x = -y + \text{sgn}(x) = -y - 1$ is negative, that is, as long as $y(t) > -1$. Once $y(t)$ crosses -1, then x will be positive, and we will have to jump back across to $x = +2$ keeping $y = -1$ constant again using the rescaled time. In retrospect, we see now that in the calculation on the right–hand branch (when $x > 0$) the correct initial condition for y is $y(0) = -1$.

This completes the calculation of the singular trajectory. Figure A.13 illustrates this. We have the following:

$$\begin{aligned} y(t) &= -1 + t, & x(t) &= 2 - t & \text{for} \quad 0 < t < 2, \\ y(t) &= 1 - (t - 2), & x(t) &= -2 + (t - 2) & \text{for} \quad 2 < t < 4, \end{aligned}$$

in the normal time coordinates. In the scaled time coordinates, $x(t)$ jumps from 0 to -2 while $y = 1$, satisfying

$$x(t) = -2(1 - \exp((t - 2)/\epsilon)),$$

and from 0 to 2 while $y = -1$, satisfying

$$x(t) = 2(1 - \exp(-(t - 4)/\epsilon)).$$

The period of the oscillation is 4 to lowest order. The function $y(t)$ is continuous along the trajectory. The complete solution for $x(t)$ over one period is

$$\begin{aligned} x(t) &= 2 - t - 2\exp(-t/\epsilon) & \text{for} \quad 0 \le t < 2, \\ x(t) &= t - 4 + 2\exp[(2 - t))/\epsilon)] & \text{for} \quad 2 \le t < 4. \end{aligned}$$

Suggested Readings

- *Mathematical Models in Biology*, Leah Edelstein-Keshet. This book details most of the mathematical techniques presented in this chapter, and contains a particularly good discussion of phase plane analysis, including linearization, stability, and qualitative analysis of systems of differential equations (Edelstein-Keshet 1988).
- *Applied Mathematics*, J. David Logan. This book covers a range of more advanced topics, in particular, perturbation and bifurcation theory (Logan 1997).

A.7 EXERCISES

1. Using manipulations comparable to those used to obtain (A.23) show that x_2 in (A.18) also satisfies the second order equation (A.23).
2. Show by substitution that if $\mathbf{x}'$ and $\mathbf{x}$ are two different solutions to (A.19), then $c'\mathbf{x}' + c\mathbf{x}$ is also a solution.
3. Show by substition into (A.35) that $c't\exp(\lambda t)$ is a second solution to (A.23) when $disc\hat{A} = 0$. [Hint: Recall that $(t\exp(\lambda t))\dot{} = (1 + \lambda t)\exp(\lambda t)$; use this to show that $(t\exp(\lambda t))\ddot{} = \lambda(2 + t)\exp(\lambda t)$.]
4. Show that any 2×2 matrix of the form $\hat{A} = \begin{pmatrix} a & b \\ 0 & a \end{pmatrix}$ with a and b arbitrary real numbers has $disc\hat{A} = 0$ and $\lambda = a$.
5. Use the solution (A.27) to the characteristic equation for $\hat{A}$, to show that $tr\hat{A} = \lambda_+ + \lambda_-$ and $det\hat{A} = \lambda_+\lambda_-$.
6. Verify that the expression for the eigenvector of $\hat{A}$ given in (A.45) is correct by multiplying that expression by $\hat{A}$. Hint: You will need to use the fact that $det\hat{A} = \lambda^2 - (a_{11} + a_{22})\lambda$, which follows from (A.27).
7. Solve the general two-variable linear equations (A.17)–(A.18) numerically: Find $x_1(t)$ and $x_2(t)$ for the matrices $\hat{A} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$, $\hat{A} = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$, and $\hat{A} = \begin{pmatrix} -2 & 1 \\ -3 & 1 \end{pmatrix}$ and $y_1 = y_2 = 0$. Determine the characteristic values of all three matrices and compare your numerical solutions to the solutions that you would expect based on the characteristic values. Explore how the solutions change when you change the values of y_1 and y_2.
8. Solve (A.17)–(A.18) numerically for the matrix $\hat{A} = \begin{pmatrix} 1 & -1 \\ 3 & 6 \end{pmatrix}$ and $y_1 = 1$ and $y_2 = 2$. Constrast your result with that in (A.41).
9. Make a phase plane plot of the solutions to the linear ODEs plotted as time series in Figure A.1A–C.
10. Show that the velocity vector for a point in phase space is parallel to the trajectory at the point. Hint: Calculate the slope of the trajectory dx_2/dx_1 using the ODEs.
11. Show that the nullclines for the general 2×2 linear equations (A.17) and (A.18) are linear.

12. Using the result in (A.47) verify the statement in Section A.4.1 that saddle point trajectories that start in the direction of the positive eigenvector grow away from the steady state exponentially, while those in the direction of the negative eigenvector approach the steady state exponentially.

13. Construct the solution to the following initial value problem, which arises in enzyme kinetics:

$$x' = 1 - xy, \quad \epsilon y' = -xy + 1 - y,$$

with the initial conditions $x(0) = 0, y(0) = 0$.

14. Find the periodic solution to

$$\epsilon x' = f(x) - y \quad y' = x,$$

where

$$f(x) = \begin{cases} -x - 2 & \text{for} \quad x < -1, \\ x & \text{for} \quad -1 \le x \le 1, \\ -x + 2 & \text{for} \quad x > 1. \end{cases}$$

15. Develop Taylor series for the following functions:
    - $\cos(t)$ around $t = 0$
    - $\ln(t)$ around $t = 1$.
    - $\exp(t^2)$ around $t = 0$. Hint: Use the exponential series we have already determined.

APPENDIX B

Solving and Analyzing Dynamical Systems Using XPPAUT

G. Bard Ermentrout

Most of the examples and exercises in the book have been designed to be solvable with the ordinary differential equations solution and analysis package XPPAUT. One reason that we emphasize the use of XPPAUT rather than one of the other available packages is that XPPAUT is distributed at no cost and runs under both Unix and Windows environments. XPPAUT will also run under the new Macintosh operating system OSX with the appropriate Xwindows server. The second reason is that XPPAUT incorporates the bifurcation package AUTO, which is not included in other packages. The Windows version of XPPAUT, Winpp, uses a different bifurcation package, as explained below.

XPPAUT can be obtained from the web site of Bard Ermentrout, the developer. The site contains instructions for the installation of XPPAUT on various platforms, as well as a very useful tutorial. The web site is:

`http://www.math.pitt.edu/~bard/xpp/xpp.html`.

In addition, there is a full–length book describing the details of XPPAUT available (Ermentrout 2002). The tutorial in this appendix will introduce the reader to the main tools available in XPPAUT that are necessary to solve most of the exercises in this book.

B.1 Basics of Solving Ordinary Differential Equations

B.1.1 Creating the ODE File

Consider the simple linear differential equation system

$$\begin{aligned} \frac{dx}{dt} &= ax + by, \\ \frac{dy}{dt} &= cx + dy, \end{aligned} \tag{B.1}$$

where a, b, c, d are parameters. We will explore the behavior of this two-dimensional system using XPPAUT (even though it is easy to obtain a closed–form solution). To analyze a differential equation using XPPAUT, you must create an input file that tells the program the names of the variables and parameters, and defines the equations. By convention, these files have the file extension `ode`, and we will call them ODE files. Here is an ODE file for system (B.1):

```
# linear2d.ode
#
# right hand sides
dx/dt=a*x+b*y
dy/dt=c*x+d*y
#
# parameters
par a=0,b=1,c=-1,d=0
#
# some initial conditions
init x=1,y=0
#
# we are done
done
```

We have included some comments indicated by lines starting with #; these are not necessary but can make the file easier to understand. The rest of the file is fairly straightforward. The values given to the parameters are optional; by default they are set to zero. The `init` statement is also optional. The minimal file for this system is

```
dx/dt=a*x+b*y
dy/dt=c*x+d*y
par a,b,c,d
done
```

In contrast to the more elaborate file, with the minimal file all parameters and intital conditions are set to zero. Use a text editor to type in the first file exactly as it is shown. Name the file `linear2d.ode` and save it. Note also that XPPAUT accepts other notation

for equations, and you should not be surprised to see the more compact version used in Appendix C or in ODE files you might find on the XPPAUT web site or in the XPPAUT user's manual. For example, the minimal file could be written

```
x'=a*x+b*y
y'=c*x+d*y
par a,b,c,d
done
```

That's it! You have written an ODE file. The minimal steps are as follows:

- Use an editor to open a text file.
- Write the differential equations in the file; one per line.
- Use the `par` statement to declare all the parameters in your system. Optionally define initial conditions with the `init` statement.
- End the file with the statement `done`.
- Save and close the file.

ODE FILE NOTES: The equation reader is case-insensitive, so that `AbC` and `abC` are treated as identical. In statements declaring initial conditions and parameters, **do not** put spaces between the variable and the "=" sign and the number. XPPAUT uses spaces as a delimiter. Always write `a=2.5` and **never** write `a = 2.5`.

B.1.2 Running the Program

Run XPPAUT by typing

```
xpp linear2d.ode
```

The name of the exacutable, here `xpp`, might be different for your system. Use the name of your executable, along with all of the desired command line options (see online help for details). (*If you are using Winpp, click on the* **Winpp** *icon; then choose the file from the file selection dialog box.*)

Six windows will appear on the screen, or they may be iconified (depending on the command line options). If any of the windows appear "dead" or blank, iconify them manually and then uniconify them. Next time run XPPAUT without the `-iconify` command line option.

Menu commands will appear like this, `Command`, and single–letter keyboard shortcuts will appear like this: **A**. *Do not use the CapsLock key*; all shortcuts are lowercase. Every command can be accessed by a series of keystrokes. To make sure key clicks are interpreted correctly, click on the title bar of the window for which the shortcut is intended.

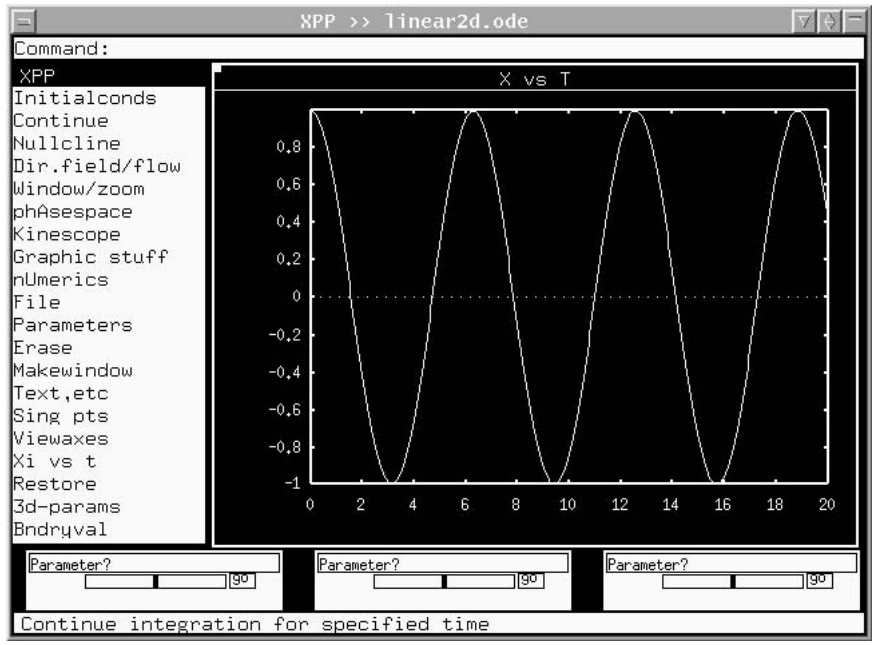

Figure B.1 The main XPPAUT window

B.1.3 The Main Window

The **Main Window** contains a large region for graphics, menus, and various other regions and buttons. It is illustrated in Figure B.1. Commands are given either by clicking on the menu items in the left column with the mouse or tapping keyboard shortcuts. After a while, as you become more used to XPPAUT, you will use the keyboard shortcuts more and more. Both the full commands and the keyboard shortcuts are included here. In general, the keyboard shortcut is the first letter of the command unless there is ambiguity (such as `Nullcline` and `nUmerics`), and then, it is just the capitalized letter (**N** and **U**, respectively). Unlike Windows keyboard shortcuts, the letter key alone is sufficient, and it is not necessary to press the Alt key at the same time. The top region of the **Main Window** is for typed input such as parameter values. The bottom of the **Main Window** displays information about various things as well as a short description of the highlighted menu item. The three little boxes with the words `parameter` are sliders to let you change parameters and initial data.

In addition to the **Main Window**, there are several other windows that appear. The **Equation Window**, shown in Figure B.2, allows you to see the differential equations that you are solving. We will describe the other windows as the tutorial progresses.

Quitting the Program

To exit XPPAUT, click `File` `Quit` `Yes` (**F** **Q** **Y**).

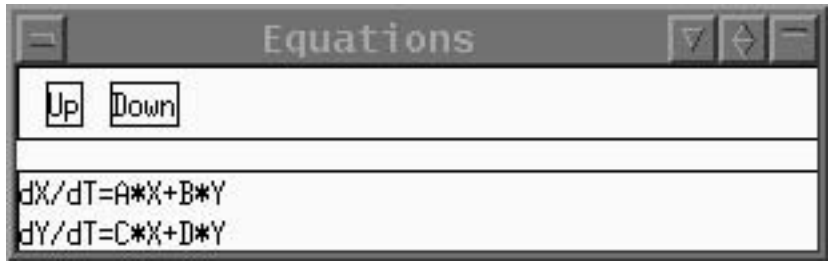

Figure B.2 The Equation Window.

B.1.4 Solving the Equations, Graphing, and Plotting.

Here, we will solve the ODEs, use the mouse to select different initial conditions, save plots of various types, and create files for printing.

Computing the Solution

In the **Main Window** you should see a box with axis numbers. The title in the window should say `X vs T`, which tells you that the variable `X` is along the vertical axis and `T` along the horizontal. The plotting range is from 0 to 20 along the horizontal and −1 to 1 along the vertical axis. When a solution is computed, this view will be shown. Click on `Init Conds` `Go` (**I** **G**) in the **Main Window**. A solution will be drawn followed by a beep. As one would expect given the differential equations, the solution looks like a few cycles of a cosine wave.

Changing the View

To plot `Y` versus `T` instead of `X`, just click on the command `Xi vs t` **X** and choose `Y` by backspacing over `X`, typing in `Y`, and typing **Enter**.

Many times you may want to plot a phase plane instead, that is, `X` vs. `Y`. To do this, click on `Viewaxes` `2D` (**V** **2**), and a dialog box will appear. Fill it in as follows:

X-axis: X	Xmax: 1
Y-axis: Y	Ymax: 1
Xmin: -1	Xlabel:
Ymin: -1	Ylabel:

Click on `OK` when you are done. (Note that you could have filled in the labels if you had wanted, but for now, there is no reason to.) You should see a nice elliptical orbit in the window. This is the solution in the phase plane (cf. Figure B.3).

Phase Plane Shortcuts

There is a very simple way to view the phase plane or view variables versus time. Look at the **Initial Data Window** (Figure B.4). You will see that there are little boxes next to the

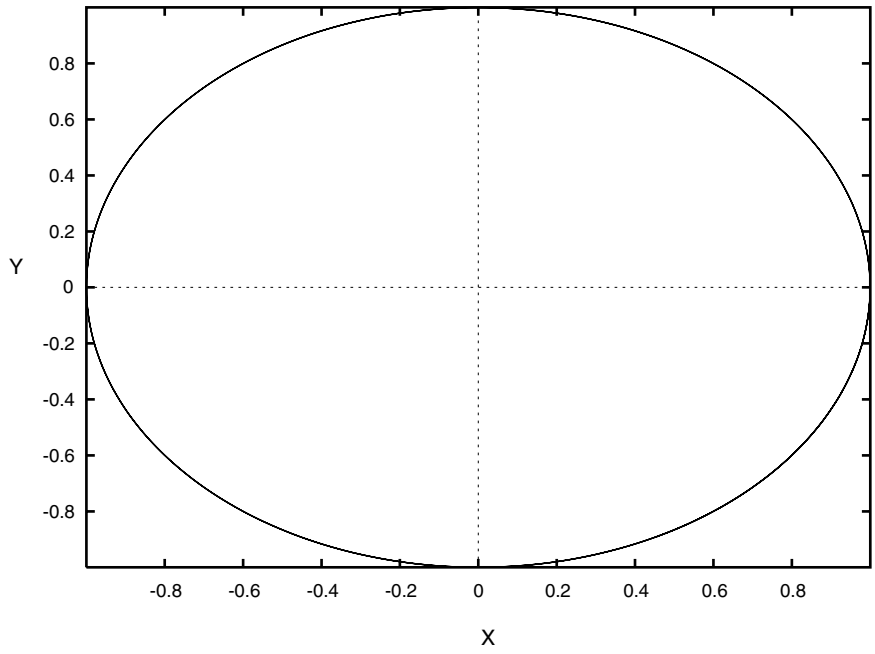

Figure B.3 Phase plane (X vs Y) for the linear 2D problem (B.1).

variable names. Check the two boxes next to `X` and `Y`. Then at the bottom of the **Initial Data Window**, click the `XvsY` button. This will plot a phase plane and automatically fit the window to contain the entire trajectory. This is a shortcut and does not give you the control that the menu command does. (For example, the window is always fit to the trajectory, and no labels are added or changed. Nor can you plot auxiliary quantities with this shortcut.) To view one or more variables against time, just check the variables you want to plot (up to 10) and click on the `XvsT` button in the **Initial Data Window**.

You should have a phase plane picture in the window. (If not, get one by following the above instructions or using the shortcut.) Click on `Init Conds` `Mouse` (**I** **M**.) Use the mouse to click somewhere in the window. You should see a new trajectory drawn. This, too, is an ellipse. Repeat this again to draw another trajectory. If you get tired of repeating this, try `Init Conds` `mIce` (**I** **I**), which, being "mice," is many mouses. Keep clicking in the window. When you are bored with this, click either outside the window or tap the escape key, **Esc**.

Click on `Erase` and then `Restore` (**E** **R**). Note that all the trajectories are gone except the latest one. XPPAUT stores only the latest one. There is a way to store many of them, but we will not explore that for now.

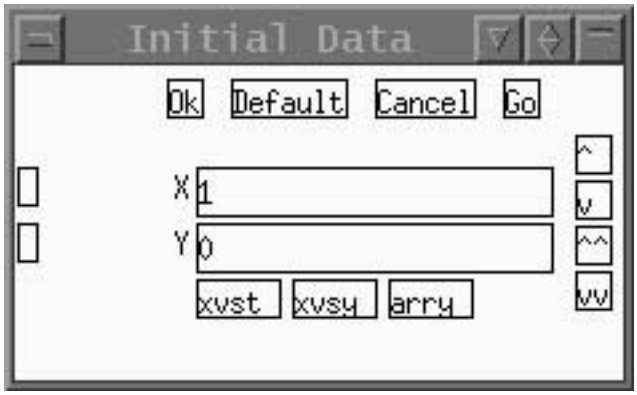

Figure B.4 The initial conditions window.

B.1.5 Saving and Printing Plots

XPPAUT does not directly send a picture to your printer. Rather, it creates a PostScript file that you can send to your printer. If you do not have PostScript capabilities, then you probably will have to use the alternative method of getting hard copy outlined below. (Note that Microsoft Word supports the import of PostScript and Encapsulated PostScript, but can only print such pictures to a PostScript printer. You can download a rather large program for Windows called GhostView which enables you to view and print PostScript on nonPostScript printers. Linux and other UNIX distributions usually have a PostScript viewer included.)

Here is how to make a PostScript file. Click on `Graphics` `Postscript` (**G** **P**), and you will be asked for three things: (i) Black and White or Color (ii) Landscape or Portrait; (iii) and the Fontsize for the axes. Accept all the defaults for now by just clicking `Enter`. Finally, you will be asked for a file name. The File Selector box is shown in Figure B.5. You can move up or down directory trees by clicking on the <>; choose files by clicking on them; scroll up or down by clicking on the up/down arrows on the left or using the arrow keys and the PageUp/PageDown keys on the keyboard; change the wild card; or type in a file name. For now, you can just click on `Ok` and a PostScript plot will be created and saved. The file will be called `linear2d.ode.ps` by default, but you can call it anything you want.

Once you have the PostScript file, you can type

```
lpr filename
```

on UNIX. In Windows, if your computer is hooked up to a PostScript printer, then you can print from a viewing application or type

```
copy filename lpt1
```

from the command line (if available).

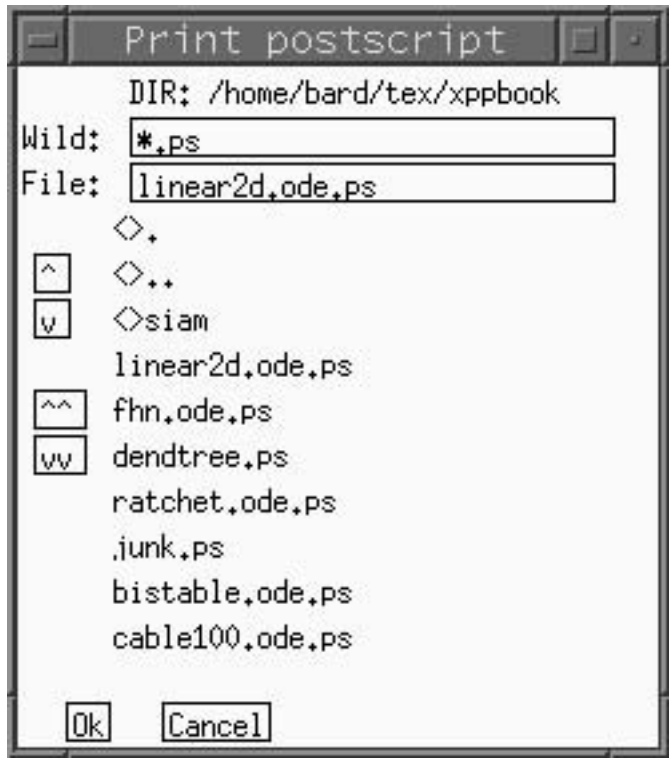

Figure B.5 File selector.

Other Ways to get Hard Copy

Another way to get hard copy that you can import into documents is to grab the image from the screen. In Windows, click on `Alt+PrtSc` after making the desired window active. Paste this into the MSPaint accessory and then use the tools in Paint to cut out what you want. Pasting into Microsoft Word is useful for generating reports with added text. Alternatively, you can download a number of programs that let you capture areas of the screen. In the UNIX environment, you can capture a window using `xv`, an excellent utility that is free and available for most UNIX versions. All of the screen shots in this tutorial were captured with `xv`. Finally, you can capture the screen (or a series of screen images) with the `Kinescope` `Capture` command and then write these to disk with the `Kinescope` `Save` command. This produces a series of GIF files that are usable by many software packages.

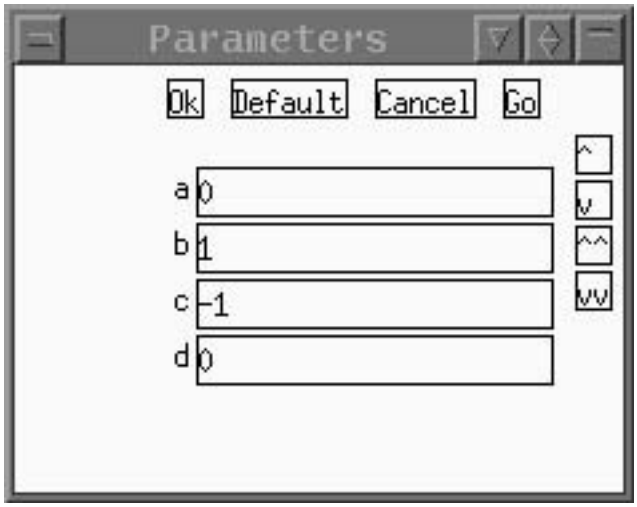

Figure B.6 The parameter window.

Getting a Good Window

If you have computed a solution and do not have a clue about the bounds of the graph, let XPPAUT do all the work. Click on `Window/zoom` `(F)it`, and the window will be resized to a perfect fit. The shortcut is **W** **F** and you will likely use it a lot!

B.1.6 Changing Parameters and Initial Data

There are many ways to vary the parameters and initial conditions in XPPAUT. We have already seen how to change the initial data using the mouse. This method works for any n-dimensional system as long as the current view is a phase plane of two variables. Here are two other ways to change the initial data:

- From the main menu click on `Init Conds` `New` and manually input the data at the prompts. You will be prompted for each variable in order. (For systems with hundreds of variables, this is not a very good way to change the data!)
- In the **Initial Data Window** you can edit the particular variable you want to change. Just click in the window next to the variable and edit the value. Then click on the `Go` button in the **Initial Data Window**. If there are many variables, you can use the little scroll buttons on the right to go up and down a line or page at a time. If you click the mouse in the text entry region for a variable, you can use the **PageUp**, etc., keys to move around. Clicking **Enter** rolls around in the displayed list of initial conditions. The `Default` button returns the initial data to those with which the program started. If you do not want to run the simulation, but have set the initial data, *you must* click on the `Ok` button in the **Initial Data Window** for the new initial data to be recognized.

  There are many ways to change parameters as well. Here are three of them:

- From the **Main Window**, click on Parameters. In the command line of the **Main Window**, you will be prompted for a parameter name. Type in the name of a parameter that you want to change. Click on Enter to change the value and Enter again to change another parameter. Click on Enter a few times to get rid of the prompt.
- In the **Parameter Window** (shown in Figure B.6) type in values next to the parameter you want to change. Use the scroll buttons or the keyboard to scroll around. As in the **Initial Data Window**, there are four buttons across the top. Click on Go to keep the values and run the simulation; click on Ok to keep the parameters without running the simulation. Click on Cancel to return to the values since you last pressed Go or Ok. The Default button returns the parameters to the values when you started the program.
- Use the little sliders (Figure B.7). We will attach the parameter d to one of the sliders. Click on one of the unused parameter sliders. Fill in the dialog box as follows:

Parameter: d
Value: 0
Low: -1
High: 1

and click Ok. You have assigned the parameter d to one of the sliders and allowed it to range between -1 and 1. Grab the little slider with the mouse and move it around. Watch how d changes. Now click on the tiny go button in the slider. The equations will be integrated. Move the slider some more and click on the go button to get another solution.

B.1.7 Looking at the Numbers: The Data Viewer

In addition to the graphs that XPPAUT produces, it also gives you access to the actual numerical values from the simulation. The **Data Viewer** shown in Figure B.8 has many buttons, some of which we will use later in the book. The main use of this is to look at the actual numbers from a simulation. The independent variable occupies the leftmost column, and the dependent variables fill in the remaining windows. Click on the top of the **Data Viewer** to make it the active window. The arrow keys and the **PageUp**,

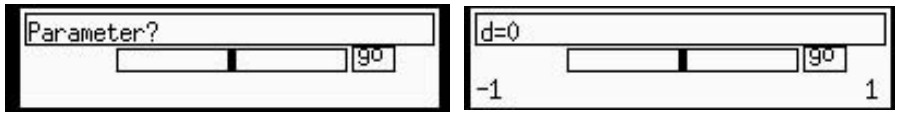

Figure B.7 Left: Unused parameter slider. Right: parameter slider used for d.

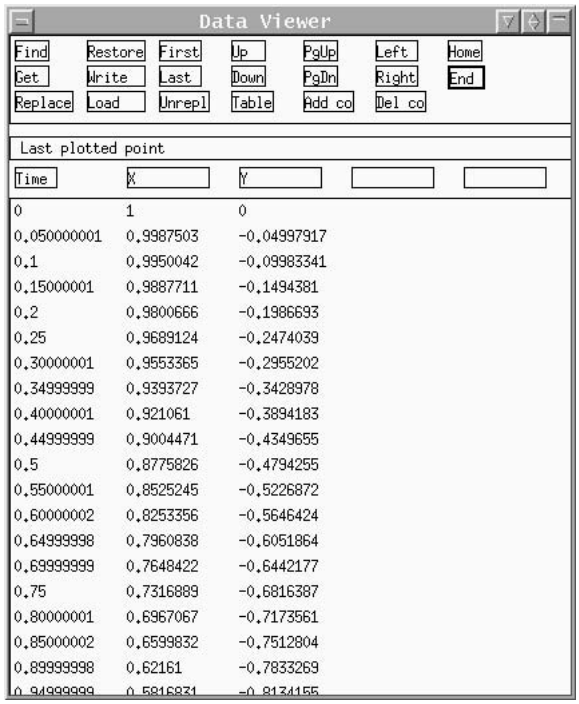

Figure B.8 The Data Viewer.

PageDown, **Home**, and **End** keys (as well as their corresponding buttons) do all the obvious things. Left and right keys scroll horizontally, a useful feature if you have many variables. Here we mention three buttons of use:

`Find` brings up a dialog box prompting you for the name of a column and a value. If you click on `Ok`, XPPAUT will find the entry that is closest and bring that row to the top. You can find the maximum and minimum, for example, of a variable.
`Get` loads the top line of the **Data Viewer** as initial data.
`Write` writes the entire contents of the browser to a text file that you specify.

B.1.8 Saving and Restoring the State of Simulations

Often you will have a view, a set of parameters, and initial data that you want to keep. You can save the current state of XPPAUT by clicking on `File` `Write set` (**F** **W**) in the **Main Window**. This brings up a file selection box. Type in a file name; the default extension is `.set`. The resulting file is an ASCII file that is human and computer readable. The first and last few lines look like this:

```
## Set file for linear2d.ode on Fri Aug  4 13:53:31 2000
2   Number of equations and auxiliaries
4   Number of parameters
```

```
# Numerical stuff
1    nout
40    nullcline mesh

.......

RHS etc ...
dX/dT=A*X+B*Y
dY/dT=C*X+D*Y
```

Once you quit XPPAUT, you can start it up again and then use the File Read set to load up the parameters, etc., that you saved.

Now you should quit the program. We will look at a nonlinear equation next, find fixed points, and draw some nullclines and direction fields. To quit, click on File Quit Yes (**F** **Q** **Y**).

B.1.9 Important Numerical Parameters

XPPAUT has many numerical routines built into it, and thus there are many numerical parameters that you can set. These will be dealt with in subsequent sections of the book where necessary. However, the most common things you will want to change are the total amount of time to integrate and the step size for integration. You may also want to change the method of integration from the default fixed–step Runge–Kutta algorithm. To alter the numerical parameters, click on nUmerics (**U**), which produces a new menu. This is a top–level menu, so you can change many things before going back to the main menu. To go back to the main menu, just click on the [Esc]-exit or tap **Esc**. There are many entries in the numerics menu. The following four are the most commonly used:

- Total sets the total amount of time to integrate the equations. (Shortcut: **T**.)
- Dt sets the size of the time step for the fixed step size integration methods and sets the output times for the adaptive integrators. (Shortcut: **D**.)
- Nout sets the number of steps to take before *plotting* an output point. Thus, to plot every fourth point, change Nout to 4. For the variable step size integrators, this should be set to 1.
- Method sets the integration method. There are currently 13 available. (Shortcut: **M**.) They are described in the user manual.

When you are done setting the numerical parameters, just click on Esc-exit or tap the Esc key.

B.1.10 Command Summary: The Basics

Initialconds | Go computes a trajectory with the initial conditions specified in the **Initial Data Window** (**I** | **G**).
Initialconds | Mouse computes a trajectory with the initial conditions specified by the mouse. Initialconds | m(I)ce lets you specify many initial conditions (**I** | **M** or **I** | **I**).
Erase erases the screen (**E**).
Restore redraws the screen (**R**).
Viewaxes | 2D lets you define a new 2D view (**V** | **2**).
Graphic stuff | Postscript allows you to create a PostScript file of the current graphics (**G** | **P**).
Kinescope | Capture allows you to capture the current view into memory, and Kinescope | Save writes this to disk.
Window/zoom | (F)it fits the window to include the entire solution (**W** | **F**).
File | Quit exits the program (**F** | **Q**).
File | Write set saves the state of XPPAUT(**F** | **R**).
File | Read set restores the state of XPPAUT from a saved .set file (**F** | **R**).

B.2 Phase Planes and Nonlinear Equations

Here we want to solve a nonlinear equation. We will choose a planar system, since there are many nice tools available for analyzing two-dimensional systems. A classic model is the FitzHugh–Nagumo equations, which are used as a model for nerve conduction. The equations are

$$\begin{aligned} \frac{dv}{dt} &= Bv(v-\beta)(\delta - v) - Cw + I_{\text{app}}, \\ \frac{dw}{dt} &= \epsilon(v - \gamma w), \end{aligned} \tag{B.2}$$

with parameters $I_{\text{app}}, B, C, \beta, \delta, \epsilon, \gamma$. Here we will use $I_{\text{app}} = 0, B = 1, C = 1, \beta = .1, \delta = 1, \gamma = 0.25$, and $\epsilon = .1$. Let us write an ODE file for this:

```
# Fitzhugh-Nagumo equations
dv/dt=B*v*(v-beta)*(delta-v)-Cw+Iapp
dw/dt=epsilon*(v-gamma*w)
par Iapp=0,B=1,C=1,beta=.1,delta=1,gamma=.25,epsilon=.1
@ xp=V,yp=w,xlo=-.25,xhi=1.25,ylo=-.5,yhi=1,total=100
@ maxstor=10000
done
```

We have already seen the first four lines: (i) lines beginning with a # are comments, (ii) the next two lines define the differential equations, and (iii) the line beginning with `par` defines the parameters and their default values. The penultimate line beginning with the `@` sign is a directive to set some of the options in XPPAUT. These could all be done within the program, but this way everything is all set up for you. Details of these options are found in the user manual. For the curious, these options set the x-axis (`xp`) to be the `v` variable, the y-axis (`yp`) to be the `w` variable, the plot range to be $[-.15, 1.25] \times [-.5, 1]$, and the total amount of integration time to be 100. The last option, `@ maxstor=10000`, is a very useful one. XPPAUT allocates enough storage to keep 4000 time points. You can make it allocate as much as you want with this option. Here we have told XPPAUT to allocate storage for 10000 points. Type this in and save it as `fhn.ode`.

B.2.1 Direction Fields

Run the file by typing `xpp fhn.ode`. The usual windows will pop up. One of the standard ways to analyze differential equations in the plane is to sketch the *direction fields*. Suppose that the differential equation is

$$\frac{dx}{dt} = f(x,y), \qquad \frac{dy}{dt} = g(x,y).$$

The phase plane is divided into a grid, and at each point (x,y) in the grid a vector is drawn with (x,y) as the base and $(x+sf(x,y), y+sg(x,y))$ as the terminal point, where s is a scaling factor. This so-called direction field gives you a hint about how trajectories move around in the plane. XPPAUT lets you quickly draw the direction field of a system. Click on `Dir.field/flow` `(D)irect Field` (**D** **D**) and then accept the default of 10 for the grid size by clicking `Enter`. A bunch of vectors will be drawn on the screen, mainly horizontal. They are horizontal because ϵ is small so that there is little change in the w variable. The length of the vectors is proportional to the magnitude of the flow at each point. At the head of each vector is a little bead. If you want to have pure direction fields that do not take into account the magnitude of the vector field, just click on `Dir.field` `(S)caled Dir. Fld` (**D** **S**) and use the default grid size. (We prefer pure direction fields, but this is a matter of taste.)

Click on `Initialconds` `m(I)ce` to experiment with a bunch of different trajectories. Note how the vectors from the direction field are tangent to the trajectories. See Figure B.9.

B.2.2 Nullclines and Fixed Points

As discussed in earlier chapters, a powerful technique for the analysis of planar differential equations and related to the direction fields is the use of *nullclines*. Nullclines are curves in the plane along which the rate of change of one or the other variable is zero. The x-nullcline is the curve where $dx/dt = 0$, that is, $f(x,y) = 0$. Similarly, the y-nullcline is the curve where $g(x,y) = 0$. The usefulness of these curves is that they break the plane

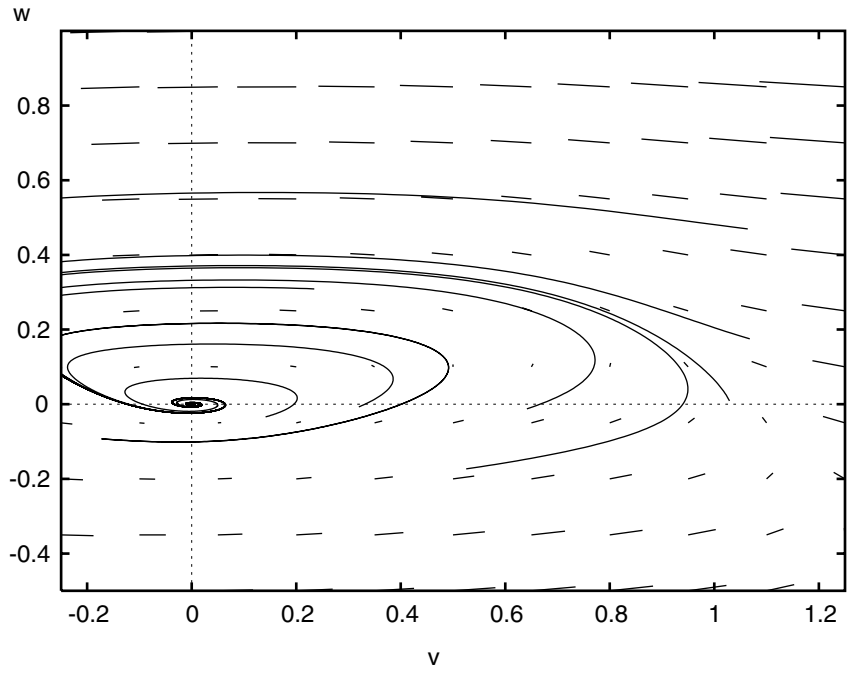

Figure B.9 Direction fields and some trajectories for the FitzHugh–Nagumo equations.

up into regions along which the derivatives of each variable have a constant sign. Thus, the general direction of the flow is easy to determine. Furthermore, any point where they intersect represents a fixed point of the differential equation.

XPPAUT can compute the nullclines for planar systems. To do this, just click on Nullcline New (N N). You should see two curves appear: a red one representing the V-nullcline and a green one representing the W-nullcline. The green one is a straight line, and the red is a cubic. They intersect just once: There is a single fixed point. Move the mouse into the phase plane area and hold it down as you move it. At the bottom of the **Main Window** you will see the x and y coordinates of the mouse. The intersection of the nullclines appears to be at $(0, 0)$.

The stability of fixed points is determined by linearizing the system of equations about them and then finding the eigenvalues of the resulting linear matrix. XPPAUT will do this for you quite easily. XPPAUT uses Newton's method to find the fixed points and then numerically linearizes the system about them to determine stability. To use Newton's method, a decent guess needs to be provided. For planar systems this is easy to do; it is just the intersection of the nullclines. In XPPAUT fixed points and their stability are found using the Sing pts command, since "singular points" is a term sometimes used for fixed points or equilibrium points. Click on Sing pts Mouse (S M) and move the mouse to near the intersction of the nullclines. Click

the button, and a message box will appear on the screen. Click on No, since we do not need the eigenvalues. A new window will appear that contains information about the fixed points. The stability is shown at the top of the window.

The nature of the eigenvalues follows: **c+** denotes the number of complex eigenvalues with positive real part; **c-** is the number of complex eigenvalues with negative real part; **im** is the number of purely imaginary eigenvalues; **r+** is the number of positive real eigenvalues; and **r-** is the number of negative real eigenvalues. Recall that a fixed point is linearly stable if all of the eigenvalues have negative real parts. Finally, the value of the fixed points is shown under the line. As can be seen from this example, there are two complex eigenvalues with negative real parts: the fixed point is $(0, 0)$. (XPPAUT reports a very small nonzero fixed point due to numerical error.) Integrate the system using the mouse, starting with initial conditions near the fixed point. (In the **Main Window**, tap **I** **I**.) Note how solutions spiral into the origin, as is expected when there are complex eigenvalues with negative real parts.

For nonplanar systems of differential equations you must provide a direct guess. Type your guess into the **Initial Data Window** and click on Ok in the **Initial Data Window**. Then from the **Main Window**, click on Sing Pts Go (**S**,G).

Change the parameter I from 0 to 0.4 in the **Parameter Window** and click on Ok in the **Parameter Window**. In the **Main Window** erase the screen and redraw the nullclines: Erase Nullclines New (**E** **N** **N**). The fixed point has moved up. Check its stability using the mouse (Sing pts Mouse). The fixed point should be $(0.1, 0.4)$. Use the mouse to choose a bunch of initial conditions in the plane. All solutions go to a nice limit cycle. That is, they converge to a closed curve in the plane representing a stable periodic solution.

We can make a nice picture that has the nullclines, the direction fields, and a few representative trajectories. Since XPPAUT keeps only the last trajectory computed, we will "freeze" the solutions we compute. We can freeze trajectories automatically or one at a time, and we will do the former. Click on Graphic stuff (F)reeze (O)n freeze (**G** **F** **O**) to permanently save computed curves. Up to 26 can be saved in any window. Frist we use the mouse to compute a bunch of trajectories. Draw the direction fields by clicking Dir.field/flow (D)irect Field (**D** **D**).

We can label the axes as follows: Click on Viewaxes 2D (**V** **2**), and the 2D view dialog will come up. Change nothing but the labels (the last two entries), and put V as the **Xlabel** and w as the **Ylabel**. Click on Ok to close the dialog. Finally, since the axes are confusing in the already busy picture, click on Graphic stuff aXes opts (**G** **X**) and in the dialog box change the 1's in the entries **X-org(1=on)** and **Y-org(1=on)** to 0's to turn off the plotting of the X and Y axes. Click Ok when you are done.

To create a PostScript file, follow Graphic stuff (P)ostscript (**G** **P**) and accept all the defaults. Name the file whatever you want and click on **Ok** in the file selection box. Figure B.10 shows the version that we made. Yours will be slightly different. If you want to play around some more, turn off the automatic freeze option,

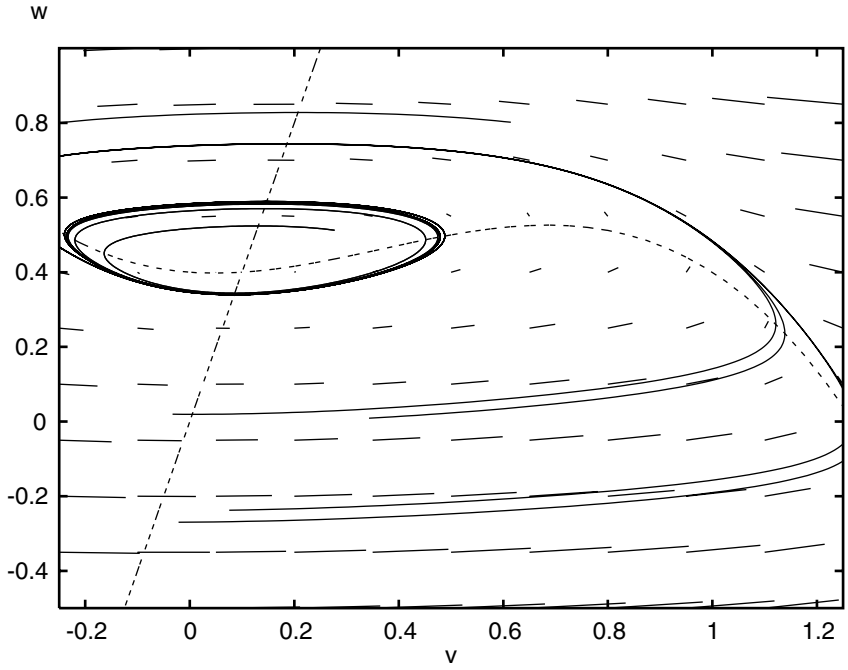

Figure B.10 Nullclines, direction fields, trajectories for `Iapp=0.4` in the Fitzhugh-Nagumo equations.

`Graphic stuff` `Freeze` `Off freeze` (**G** **F** `O`), and delete all the frozen curves, `Graphic stuff` `Freeze` `Remove all` (**G** **F** **R**).

B.2.3 Command Summary: Phase Planes and Fixed Points

`Nullcline` `New` draws nullclines for a planar system (**N** **N**).

`Dir.field/flow` `(D)irect Field` draws direction fields for a planar system (**D** **D**).

`Sing pts` `Mouse` computes fixed points for a system with initial guess specified by the mouse (**S** **M**).

`Sing pts` `Go` computes fixed points for a system with initial guess specified by the current initial conditions (**S** **G**).

`Graphic stuff` `Freeze` `On Freeze` will permanently keep computed trajectories in the current window (**G** **F** **O**).

`Graphic stuff` `Freeze` `Off Freeze` will toggle off the above option (**G** **F** **O**).

`Graphic stuff` `Freeze` `Remove all` deletes all the permanently stored curves (**G** **F** **R**).

`Graphic stuff` `aXes opts` lets you change the axes (**G** **X**).

`Viewaxes` `2D` allows you to change the 2D view of the current graphics window and to label the axes (**V** **2D**).

B.3 Bifurcation and Continuation

Once we have found the critical points of a system of interest, we can then embark on a continuation and bifurcation analysis of the solutions, as we mentioned in Appendix A. Continuation analysis describes how solutions to differential equations evolve over parameters, while bifurcation analysis refers, in particular, to how solutions appear and disappear as parameters are varied. One of the main strengths of XPPAUT is that it provides a convenient interface to many of the features found in the AUTO package for continuation/bifurcation analysis. AUTO remains one of the best such packages. However, the stand-alone versions of AUTO require coding compilation of the equations with the FORTRAN computer language. Note that AUTO currently only is available for the Unix version of XPPAUT. The Windows version uses a continuation package called LOCBIF, and slight differences from the procedures outlined below are explained in the user's manual on the web site. While AUTO is powerful even as implemented in XPPAUT, it is not foolproof. The results you obtain should be viewed with a critical eye. Bifurcation analysis is discussed in more depth in an excellent book by Kuznetsov (Kuznetsov 1998). It is useful to understand that the AUTO features available in XPPAUT are independent of the other tools, but that parameters and fixed points are exchanged back and forth. Bifurcation diagrams can be imported into XPPAUT for plotting.

B.3.1 General Steps for Bifurcation Analysis

- AUTO bifurcation analysis must start from a fixed (or singular) point or from a periodic orbit. Get rid of transients and find a stable fixed point by by integrating several times: Click `Initialconds` `Go` and then click `Initialconds` `Last` several times. For limit cycles, get a good estimate of the period and integrate one full period only.
- Bring up the **AUTO window** by selecting `File` `Auto` (**F** **A**).
- Use the `Parameter` function to choose the parameters that will be varied.
- Use the `Axes` `Hilo` function to select the parameters to be plotted and the range over which they will be varied. A one–parameter bifurcation diagram must be completed before a second parameter can be varied.
- Use the `Numerics` function to define direction, step size, etc.

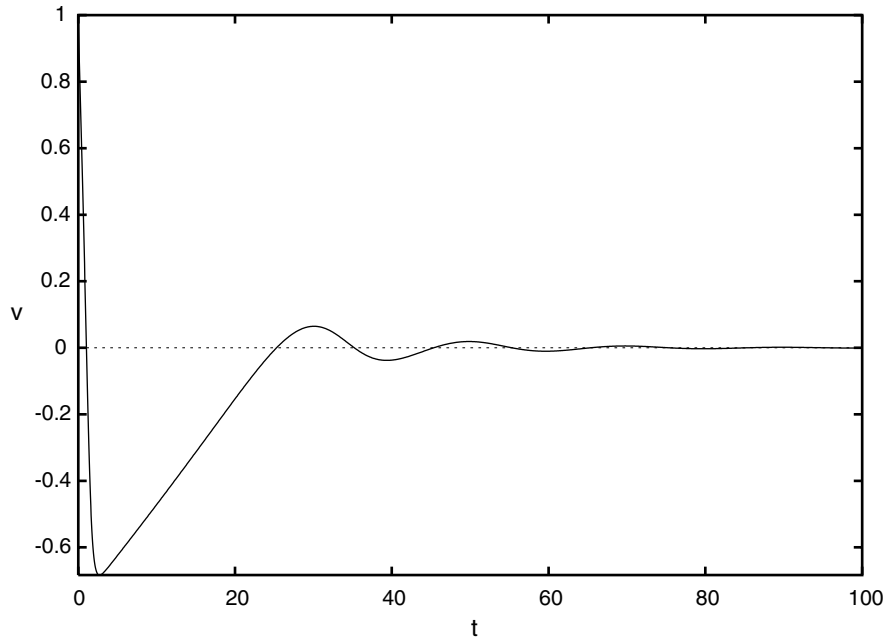

Figure B.11 FH-N system with `Iapp=0`.

- `Run` the analysis for `Steady state` or `Periodic`.

It is more difficult for AUTO to start from a periodic solution, and so some further assistance is required. First, a good approximation of the period must be determined from the data browser (use `Find` in conjunction with a large number to determine the maximum) or by measuring the peak-to-peak period with the mouse. Integrate over just that period by adjusting `nUmerics` `Total` to the period length. After starting AUTO and selecting parameters and bounds, choose `Run` `Periodic`. AUTO may still fail, and further adjustments to numerical parameters or a better approximation of the period may be necessary.

B.3.2 Hopf Bifurcation in the FitzHugh–Nagumo Equations

We will continue with the FitzHugh-Nagumo example and explore the bifurcation structure of this system. Using the FH-N equations from earlier, add the lines

```
v(0)=1,
w(0)=1,
```

to the ODE file to set initial conditions. Now we are ready to begin the analysis:

1. Start up XPPAUT with `Iapp=0`. Next, run the ODE file, ploting `x vs t` as discussed above. You should see v oscillate a bit and go to zero, as seen in Figure B.11. Now

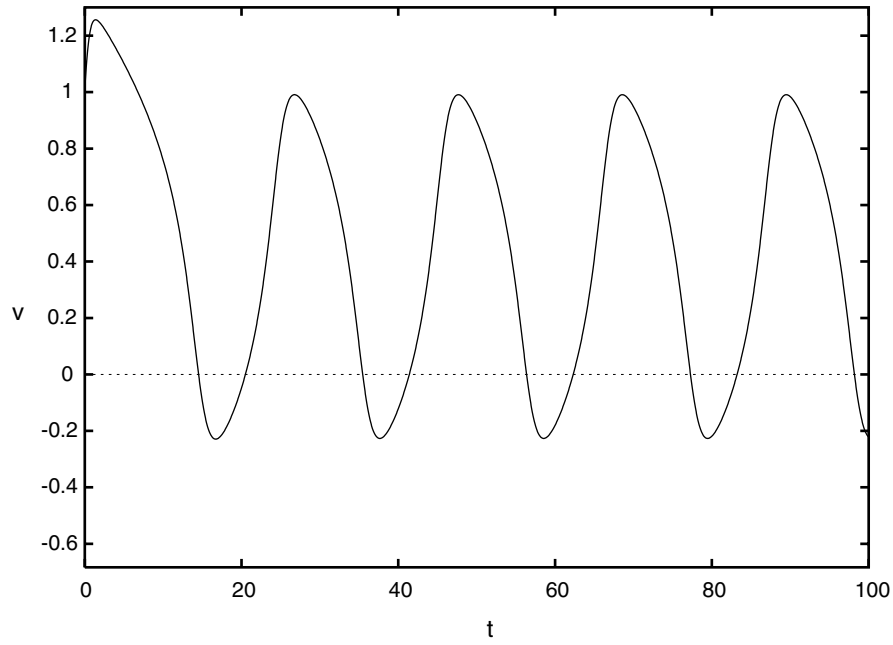

Figure B.12 FH-N system with `Iapp=1.5`.

run the simulation with `Iapp=1.5`. You should see the periodic solution shown in Figure B.12. We want to understand how this change ocurrs as the parameter `Iapp` is changed.

2. To set up the bifurcation analysis run the simulation again with `Iapp=0`: Click `Initialconds` `Go` and then click `Initialconds` `Last` several times. This will run the simulation until it is really at the steady state.
3. The next step is to bring up the **AUTO window** by selecting `File` `Auto` (**F** **A**). Once the **AUTO window** is present, make sure that `Iapp` is listed as `Par1` under `Parameter`. It should already be there if you typed the file in as written. If not, select `Iapp`.
4. Set up the graphics axes with `Axes` `Hilo`. Fill in the dialog box as follows:

`Y-axis:` V
`Main Parm:` Iapp
`Xmin:` -0.5
`Ymin:` -3
`Xmax:` 0.5
`Ymax:` 0.5

5. Set up the `Numerics`, and change only `Par Min=0` and `Par Max=3.5`.

6. To begin, click Run | Steady state. The beginning of the diagram should appear with four points labeled as in Figure B.13. The bold line represents a stable steady state, and the faint line represtents an unstable steady state.
7. Now choose Grab. This will allow you to see what the four marked points in the diagram represent. A cross appears on the plot at marker 1, and below the plot, a description will be present. Lab is the label for the point (1), Ty is the type of point. EP stands for End point, where we started computing. Iapp is the value of the parameter for that location of the graph.
8. Move the cursor over the (2). Under Ty should be the label HP, for Hopf bifurcation. Point (3) should also be a Hopf bifurcation, and (4) will be the other end point.
9. With the cross back on (2), press **Enter**. This "grabs" that point as the beginning of a new calculation. Click Run again. This time, the pop-up screen is labeled as Hopf Pt, and we will choose Periodic to follow the periodic orbits as Iapp changes. You should get something like Figure B.14. The darkened circles show that the periodic orbits are stable. Open circles represent an unstable periodic solution. Note that there are upper and lower points in the plot for the periodic solution, showing the maximum and minimum values (of v) that the solution attains.
10. By using grab again, we can go to the periodic orbits, and their period will be shown below the plot.
11. We can save the plot using the File menu as discussed above.

B.3.3 Hints for Computing Complete Bifurcation Diagrams

- Be sure to start at a fixed point or clearly defined limit cycle. As discussed above, get rid of transients and find a stable fixed point by integrating several times: Click

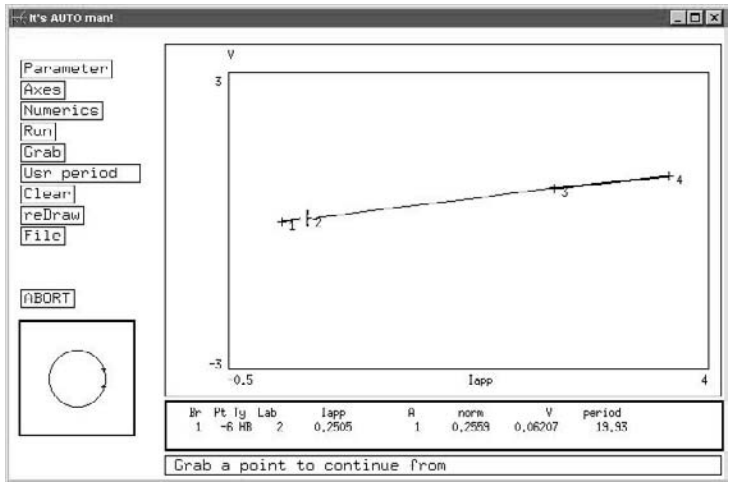

Figure B.13 Initial bifurcation diagram and AUTO window for the FH-N system.

Initialconds | Go and then Initialconds | Last several times. For limit cycles, get a good estimate of the period and integrate one full period only.

- Learn to navigate the diagram efficiently. Tab jumps to special points, Axes | Fit | reDraw will fit the entire diagram to the page, and Axes | Zoom magnifies a given area.
- AUTO will try to follow all branches of fixed points. However, AUTO may need some assistance. Grab special points and Run in different directions by changing the sign of **Ds** in the numerics dialog box.
- Try to find the periodic solution from all Hopf points.
- Be sure also to change **Ds** for two parameter bifurcations.
- An initial **MX** label indicates that auto has failed and you may not have provided a good fixed point or periodic orbit.
- To erase the diagram and start again with different parameters, Grab the initial starting point and destroy the diagram with File | Reset diagram.
- If AUTO fails to continue, try making **dsmin** smaller; for periodic orbits and boudary value problems make **ntst** larger.
- If AUTO clearly misses a bifurcation point, make **dsmin** smaller and recompute.

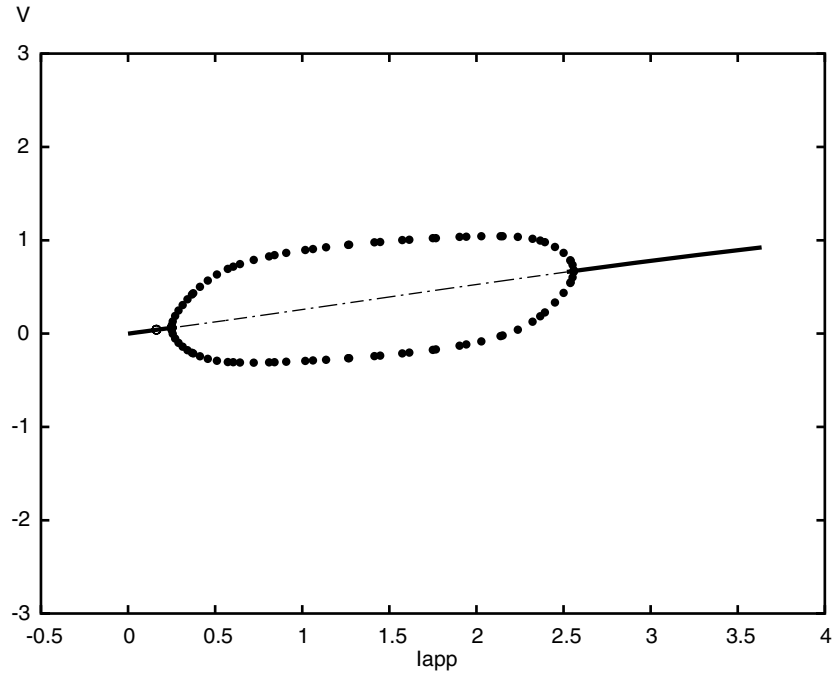

Figure B.14 Final bifurcation diagram the FH-N system.

B.4 Partial Differential Equations: The Method of Lines

XPPAUT doesn't have any way to solve PDEs other than by discretizing space and producing a series of ODEs using the method of lines. However, one does not have to write all the differential equations down, one at a time. There are ODE file shortcuts that make this easy to do. There is also a nice way of plotting the space–time behavior of a one-dimensional PDE. We will go through one quick example here. Consider the PDE

$$\frac{\partial v}{\partial t} = f(v,w) + D\frac{\partial^2 v}{\partial x^2},$$
$$\frac{\partial w}{\partial t} = g(v,w),$$

where f, g are the kinetics for the FitzHugh–Nagumo model or some other model. For simplicity, we assume Neumann boundary conditions (see Chapter 7). This system can be discretized with the method of lines, yielding the following system of ODEs:

$$\frac{dv_0}{dt} = f(v_0,w_0) + D(v_1 - v_0),$$
$$\frac{dv_j}{dt} = f(v_j,w_j) + D(v_{j+1} - 2v_j + vj - 1), \quad j = 1,\ldots,N-1,$$
$$\frac{dv_N}{dt} = f(v_N,w_N) + D(v_{N-1} - v_N),$$
$$\frac{dw_j}{dt} = g(v_j,w_j), \quad j = 0,\ldots,N.$$

We will use the FitzHugh–Nagumo kinetics and make an ODE file of the discretized system. Note that we have changed the parameter `B=4`, so that the system is more excitable (see Chapter 7):

```
# fitzhugh-nagumo action potential
f(v,w)=B*v*(v-beta)*(delta-v)-Cw+Iapp
g(v,w)=epsilon*(v-gamma*w)
par Iapp=0,B=1,C=1,beta=.1,delta=1,gamma=.25,epsilon=.1
par D=.5
dv0/dt=f(v0,w0)+D*(v1-v0)
dv[1..49]/dt=f(v[j],w[j])+D*(v[j+1]-2*v[j]+v[j-1])
dv50/dt=f(v50,w50)+D*(v49-v50)
dw[0..50]/dt=g(v[j],w[j])
@ total=200,dt=.25,meth=qualrk,tol=1e-6
@ xhi=200,yp=v20
done
```

XPPAUT actually expands this to 100 differential equations, and the variables are named `v0,w0,v1,w1` and so on up to `v50,w50`. You must always use the letter "j" for the index. We have told XPPAUT to use a quality step size (adaptive) Runge–Kutta

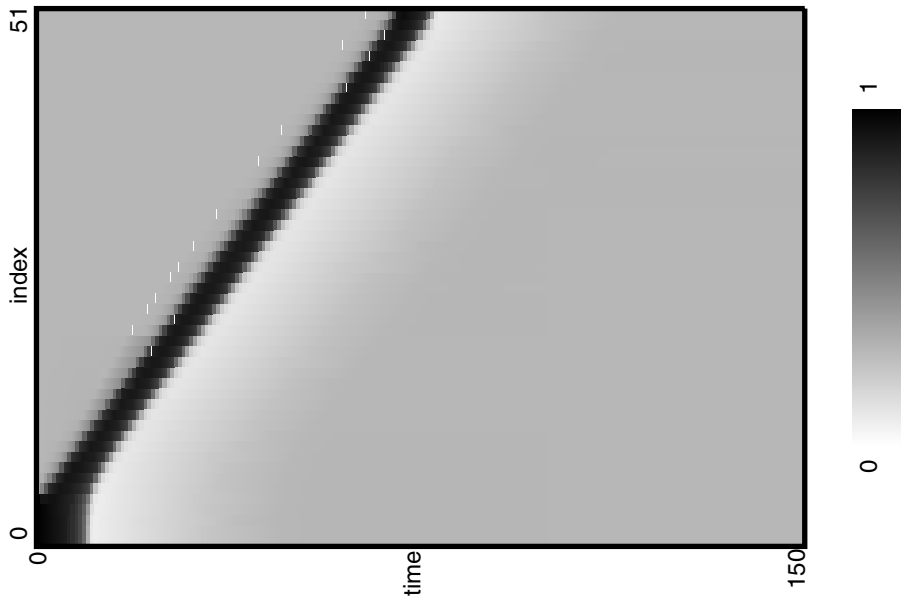

Figure B.15 XPPAUT output of the spatial FitzHugh–Nagumo simulation.

routine with an output step size of 0.25. We integrate the equations for 200 time units. We plot `v20` so that the appearance of an action potential will be clear.

Run XPPAUT with this file. Now we will give some initial conditions. Rather than type them in one by one, we will define $V_j(0)$ by a formula. Click on `Initial conds` `formUla`. When prompted for the variable type in `v[0..50]` and type in `heav(5-[j])` for the formula. Then tap the Enter key a few times. Note that the index is referred to as `[j]` in the formula rather than just `j`. You should see an action potential appear on the screen. Click on `Graphic stuff` `Add curve` and choose `V40` for the y-axis and color 7 (green) for the color. The potentials of the 20th and 40th points will appear. In the **Initial Data Window**, click on the box next to `V0`. Scroll down and click on the box next to `V50`. Now click on the button labeled `arry`. A new window will appear with the space–time plot of the potential that should look something like Figure B.15. You can fool around with the parameters for this plot by clicking on the edit box in the window.

As a last bit of analysis, we can look at the spatial profile at a fixed point in time. To do this, we will transpose all the space–time data so that the 51 columns of the potential at a particular point in time become 51 rows in the second column; the first column will hold the indices. Click on `File` Transpose. Edit the dialog box so that `NCols=51` (the number of columns); `Row 1=300` (the output time step is 0.25, so row 300 represents $t = 75$). Click `OK` to complete the transpose. (You can undo this by clicking on `File` `Transpose` and then clicking `Cancel` in the dialog box.) Once you have transposed the data, just plot `V0` versus time. This is the spatial profile at $t = 75$.

B.5 Stochastic Equations

B.5.1 A Simple Brownian Ratchet

XPPAUT has many features useful for stochastic modeling. In particular, it can simulate Brownian motion and continuous Markov processes. Before turning to a sodium channel simulation, we first create an XPPAUT file for a ratchet that moves between -1 and 1 and is not allowed to exit the boundaries. It is easiest to treat this in discrete time. The function `normal(0,1)` produces a normally distributed random number with mean 0 and standard deviation 1. Thus, the exercise can be written as the following simple ODE file:

```
par f1=-5,f2=5,h=.1,q=2
@ total=1000,meth=discrete
init x=-1
xp=x+h*.5*(f1*(sign(-x)+sign(x+1))+f2*(sign(x)+sign(1-x)))
+sqrt(q*h)*normal(0,1)
dx/dt=max(min(xp,1),-1)
done
```

The statement `meth=discrete` tells XPPAUT to treat this as a map rather than a continuous differential equation. In this mode, `xp` is the new value of `x` under the random dynamics. However we do not want x to escape the boundaries of ± 1, so when we update the new value of `x` it is constrained by the function `max(min(xp,1),-1)`.

B.5.2 A Sodium Channel Model

In the next example we simulate a sodium channel model due to Joe Patlak. The functions $\alpha_m, \beta_m, \alpha_h$ are the usual voltage dependendent functions for the Hodgkin–Huxley equations. XPPAUT can simulate a multi–state Markov process by defining a "Markov" variable (which has N states, $0, 1, \ldots, N-1$) and the transition matrix. Each row of the transition matrix is given on a single line following the declaration of the Markov variable. Each entry is contained within the curly braces, { and }. For example, suppose that you had a two state process with transition rates a from 0 to 1 and b from 1 to 0. Then you would write

```
markov z 2
{0} {a}
{b} {0}
```

Note that you can put anything you want in the diagonals, since they are ignored. Here is a complete ODE file for the above process:

```
# two state markov model
par a=.2,b=.3
```

```
markov z 2
{0} {a}
{b} {0}
@ total=50,xhi=50,xp=z,yp=z,yhi=1.5,ylo=-.5
# ddummy/dt=0
done
```

The last line should be uncommented if your version of XPPAUT does not accept this file. Older versions require at least one differential equation. Run this and integrate the equations. See z flip up and down.

Now with this trivial example in mind, we turn to the sodium channel model. Here is the XPPAUT file:

```
# model for the hh Na channel
# due to patlack
#
par vhold=-100,vnew=10
par ton=1,toff=16,ena=50
par k1=.24,k2=.4,k3=1.5
v=vhold+heav(t-ton)*heav(toff-t)*(vnew-vhold)
am=.1*(v+40)/(1-exp(-(v+40)/10))
bm=4*exp(-(v+65)/18)
ah=.07*exp(-(v+65)/20)
markov z 5
{0} {3*am} {0} {0} {0}
{bm} {0} {2*am} {0} {k1}
{0} {2*bm} {0} {am} {k2}
{0} {0} {3*bm} {0} {k3}
{0} {0} {0} {0} {ah}
aux cond=(z==3)
aux ina=(z==3)*(v-ena)/40
aux pot=v
@ meth=euler,dt=.01,total=16
@ yp=ina,ylo=-1.5,xlo=0,xhi=15,bound=100
done
```

The voltage is stepped from a value `vhold` to `vnew`. The Markov process has five states. Only state 3 is conducting. Thus, the auxiliary variable `cond=(z==3)` is zero if the channel is not conducting and 1 if it is. The current passed is given by the variable `ina`. Euler's method is used for this calculation, since that is usually the best method to use for any stochastic model. Integrate this a few times to see the channel open transiently. This is a transient channel, so even at high potentials it stays on only briefly. The Hodgkin–Huxley equations arise by assuming that there are many independent channels. Since they are assumed to be independent, we can simulate the effect of m

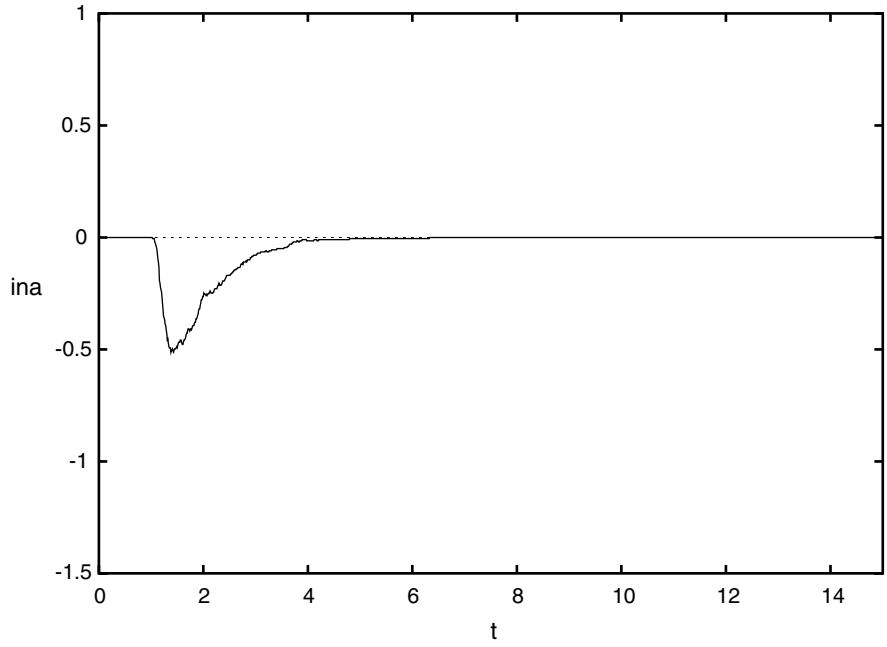

Figure B.16 XPPAUT results for the sodium channel simulation.

channels by just integrating the equations m times and averaging the output. XPPAUT does this for you. Click on nUmerics Stochastics Compute to tell XPPAUT how many trials. Choose z as the variable to range over, 200 steps with Start=0 and End=0. Then click Ok. The equations will be integrated 200 times; this is equivalent to having 200 independent channels. Once XPPAUT has done this (you can keep track by looking at the bottom), then click on stocHastics Mean to load the data browser with the mean values of all its rows over the 200 trials. Click on Escape to get back to the main menu and then click on Restore to see the mean value shown in Figure B.16. This looks very similar to the deterministic solution. Try simulating fewer channels (e.g., 10) and more channels, (e.g., 1000). Why do we have to take such small steps?

B.5.3 A Flashing Ratchet

As a final example of a stochastic equation, we simulate the "flashing ratchet" model: an asymmetric ratchet that flashes on and off at a particular rate according to a Markov process and subject to simple delta-correlated noise. Here is the model:

$$dx = -zf(x)dt + \sigma d\xi,$$

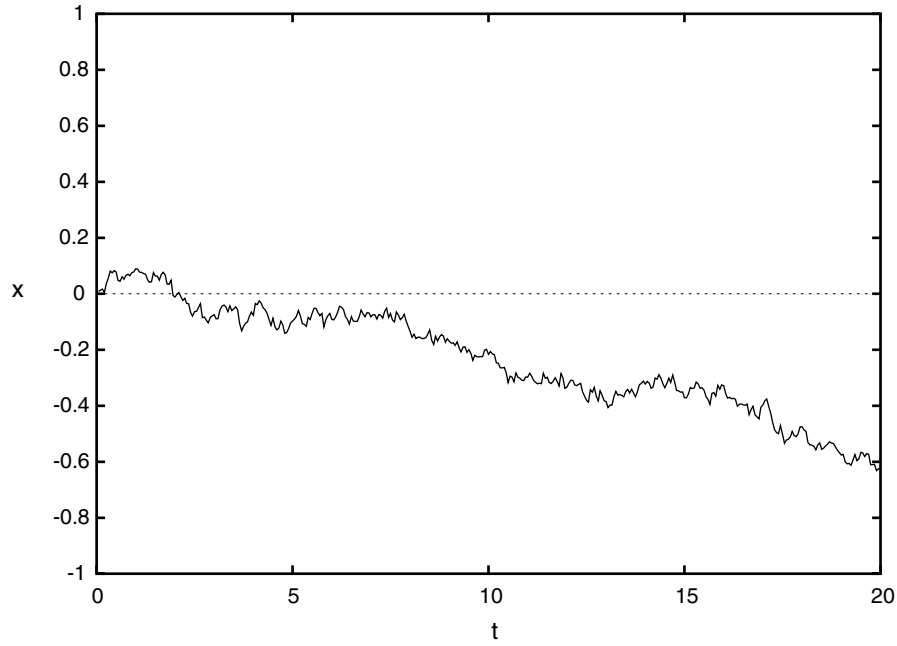

Figure B.17 Stochastic simulation of a simple Brownian ratchet.

where z is a two–state Markov process either off (0) or on (1) and f is just the derivative of an asymmetric potential. The potential should be made periodic. Here is the XPPAUT file:

```
# ratchet
#  -1 for 0<x<1
#   1/a for 1<x<1+a
ff(x)=if(x<1)then(-1)else(1/a)
f(x)=ff(mod(x,1+a))
par a=.25
par alpha=.2,beta=.2
par sig=.5
wiener xi
dx/dt=f(x)*z+sig*xi
markov z 2
{0} {alpha}
{beta} {0}
@ total=200,bound=200,meth=euler
done
```

We first define the potential on the interval $[0, 1+a)$ and then extend it to the whole line modulo $1+a$. We define a two-state markov process that flips randomly between 0 and 1. The `wiener w` declaration tells XPPAUT that this is a normally distributed number scaled by the internal time step `dt`. Thus, if you change `dt`, the standard deviation is scaled accordingly by $\sqrt{dt}$. Integrate the equations a few times. Next, do 50 simulations and look at the mean trajectory. (Use `nUmerics` `stocHastic` `Compute` as was done in the channel model.) You will see in Figure B.17 that there is persistent downward drift. This is what is predicted by theory.

APPENDIX C

Numerical Algorithms

This appendix contains samples of the different kinds of numerical constructs used to produce the figures in this book. Each of the files produces a figure in the book and is labeled accordingly. In general, each of the files contains a new construct not previously used. However, several are included for easy comparison to one that is very similar. We provide a one-line description of the intent of the code. It will be useful to look back at the figures these files produce as well.

The files presented here are working XPPAUT code. XPPAUT uses concise straightforward code, making it easy to translate to other languages and packages. The code as written can be saved as an `.ode` file and run using XPPAUT or Winpp. The code is commented heavily at the beginning and less so at the end, where the reader will be able to pick up on the more obvious features.

This appendix provides a good overview, and it provides those with programming experience a template for writing their own code. Furthermore, XPPAUT code for every computational figure in the book is available for download on the web site for the book, including the corresponding set files. As readers solve examples and exercises using other packages such as Berkeley Madonna, MATLAB, and Mathematica. These files will be included on the web site if they are submitted to us.

Figure 1.10. This is the first basic code to run. Once you have a simple XPPAUT model running, it is straightforward to modify it to increase its complexity.

```
#solves single channel model
#this is basically the same equation with the steady
#state level changed to ninf rather than zero
#the initial condition on the open fraction
fo(0)=1
#parameters
#finf is the steady state value
#tau is the the time scale
param finf=0.5,tau=3
#differential equation
dfo/dt=-(fo-finf)/tau
#auxiliary function
#rate is the instantaneous rate of change
aux rate=-(fo-finf)/tau
done
```

Figure 2.6. This program provides a template for plotting functions with XPPAUT using the aux functionality.

```
#.ode file for plotting functions
#x is the dummy variable for plotting the auxiliary functions
#initial value of x
x(0)=-80
#differential equation
#the solution to this equation is x=t+x(0)
#it lets us plot functions f(x)
dx/dt=1
#equilibrium activation and inactivation
#auxillary functions to plot
aux finfact=1/(exp(-(x+25)/5)+1)
aux finfinact=1/(exp((x+50)/2)+1)
#the characteristic times to plot
aux tauact=5/cosh((x+25)/2*5)
aux tauinact=5/cosh(-(x+50)/2*2)
done
```

Figures 2.9, 2.10, 2.11, 2.12. This is a simple two-variable model (the Morris–Lecar model) that includes function definitions outside of the differential equations.

```
#simulation of the Morris-Lecar equations
dv/dt = (-gca*minf*(V-Vca)-gk*w*(V-VK)-gl*(V-Vl)+Iapp)/C
dw/dt = phi*(winf-w)/tauw
```

```
#initial conditions
v(0)=-60.855
w(0)=0.014915
#functional forms for the equations
#keep the functions separate from the differential equations makes
#the differential equations less complicated to sort through,
#but is not necessary.
minf=0.5*(1+tanh((v-v1)/v2))
winf=0.5*(1+tanh((v-v3)/v4))
tauw=1/cosh((v-v3)/(2*v4))
#parameters
param Iapp=0,vk=-84,vl=-60,vca=120
param gk=8,gl=2,C=20
param v1=-1.2,v2=18
param v3=2,v4=30,phi=.04,gca=4.4
#numerical parameters:
#these are sometimes included to avoid needing to
#set them when the program is run
#total is the total time the simulation will run
#dt is the time step
#the rest of these parameters define the view in the graph
@ total=150,dt=0.25,ylo=-75,yhi=45,xlo=0,xhi=150,xp=t,yp=va=4.4
done
```

Figures 5.8, 5.10. This file shows that the ordering of the program elements in XPPAUT is not crucial.

```
#Keizer/Levine: reduced open-cell model
#Parameter values slightly modified from original paper.
#Two figs generated:
#(1) oscillating time series: jin=1.5 for 60<t<400 and jin=0.3 otherwise
#(2) bifurcation plot:   jin=0.1 to jin=4.5
#numerical parameters
@ meth=cvode, dtmax=1, dt=0.05, total=400, maxstor=1000000
@ bounds=100000000, xp=w,  yp=w, toler=1.0e-6, atoler=1.0e-6
@ xlo=0, xhi=400, ylo=0, yhi=2.5
#initial conditions
CAI(0)=0.2, CATOT(0)=1.
#parameters
params fi=0.01,Kserca=0.2
params Ka=0.4,Kb=0.6,Kc=0.1,kcm=0.1
params vpmca=5, Kpmca=0.6, jin=1.5
params vserca=100,vryr=5,vleak=0.2,sigma=0.02
```

```
#function definitions
winf=(1+(Ka/CAI)^4+(CAI/Kb)^3)/(1+(1/Kc)+(Ka/CAI)^4+(CAI/Kb)^3)
CAER=(CATOT-CAI)/sigma
tau=winf/kcm
Pryr=winf*(1+(CAI/Kb)^3)/(1+(Ka/CAI)^4 + (CAI/Kb)^3)
#differential equations
dCAI/dt=fi*((vryr*Pryr+vleak)*(CAER-CAI)-vserca*(CAI^2/(CAI^2+Kserca^2))\
    -vpmca*CAI^2/(CAI^2+Kpmca^2)+jin)
dCATOT/dt=fi*(jin-vpmca*CAI^2/(CAI^2+Kpmca^2))
aux Pryr=winf*(1+(CAI/Kb)^3)/(1+(Ka/CAI)^4 + (CAI/Kb)^3)
done
```

Figures 5.14, 5.15. Note here that the keyword for parameters is flexible.

```
# Li-Rinzel open-cell model .ode file.
# Dimensional version
# The equations
dCAI/dt=fi/Vi*((L+Pip3*((IP3*CAI*h)/((IP3+Ki)*(CAI+Ka)))^3)*(CAER-CAI)\
  -Vserca*CAI^2/(Kserca^2+CAI^2)+epsilon*(Jin-Vpmca*CAI^2/(Kpmca*Kpmca+CAI^2)))
dh/dt=A*(Kd-(CAI+Kd)*h)
dCATOT/dt=fi/Vi*epsilon*(Jin-Vpmca*CAI^2/(Kpmca^2+CAI^2))
CAER=(CATOT-CAI)/sigma
# The parameters
# Jin=aMol/s
par Jin=1200,fi=0.01
# Vi=pL
par Vi=4
# L,Pip3=pL/s
par L=0.37,Pip3=26640
# IP3,CAI,CAER,CATOT,Ki,Ka,Kserca,Kd,Kpmca=uM
par IP3=0.9,Ki=1.0,Ka=0.4
# Vserca, Vp=aMol/s [sic]
par Vserca=400,Kserca=0.2,A=0.5,Kd=0.4
# sigma,epsilon,fi=unitless
par sigma=0.185,epsilon=0.01,Vpmca=2000,Kpmca=0.3
# The initial conditions
CAI(0)=0.2
h(0)=0.8
CATOT(0)=4.0
aux CAER=CAER
#parameters for graphing
@ TOTAL=300,DT=0.02,xlo=0,xhi=2,ylo=0,yhi=1.5,MAXSTOR=20000
@ xplot=t,yplot=CAI
#parameters for AUTO bifurcation analysis
```

```
@ dsmin=1e-5,dsmax=.1,parmin=-.5,parmax=.5,autoxmin=-.5,autoxmax=.5
@ autoymax=.4,autoymin=-.5
set vvst {xplot=c,yplot=h,xlo=0,xhi=1.5,ylo=0,yhi=1.5,total=100,\
dt=0.01,meth=qualrk}
done
```

Figures 5.19, 5.20. This simulation exhibits bursting behavior in the right parameter range. Note here an alternative method for declaring initial conditions.

```
#Morris-Lecar-like beta-cell
init V=-65.0, n=0.00016, CAI=0.2
# equations
dV/dt=(gL*(Vl-V)+gK*n*(Vk-V)-Ica-Ikca+Iapp)/Cm
dn/dt=(ninf-n)/tau
dCAI/dt=f*(-alpha*Ica - vlpm*CAI)
# where
minf=0.5*(1+tanh((v-v1)/v2))
ninf=0.5*(1+tanh((v-v3)/v4))
tau=1/(phi*cosh((v-v3)/(2*v4)))
# For bifurcation diagram, set auto=1 and use gkcastar
# as the bifurcation parameter:
param gkcastar=100.0, auto=0
param Vk=-75, Vl=-75, Vca=25
param Iapp=0, gK=2700, gL=150, gCa=1000, Cm=5300
param v1=-20.0, v2=24, v3=-16, v4=11.2, phi=0.035
param gkca=2000.0, Kkca=5.0
# Ikca
Ikca = auto*gkcastar*(V-Vk)+((1-auto)*gkca*CAI/(Kkca+CAI))*(V-Vk)
# Calcium Handling
par alpha=4.50e-6, vlpm=0.15, f=0.001
# Ikatp
par gkatp=0.0
Ica = gca*minf*(V-Vca)
@ meth=cvode, atol=1.0e-6, tol=1.0e-6, dt=10.0
@ total=40000, maxstor=10000
@ xp=t, yp=v, bound=100000000
@ xlo=0, xhi=40000, ylo=-70, yhi=-10
done
```

Figures 6.2, 6.3. This file provides a template for simple coupled dynamics. Note that by specifying the argument explicitly, functions (such as minf(v) here) need not be duplicated. Rather, one passes the appropriate variable as in the differential equation for V1.

```
#mlgap.ode
```

```
#gap-junction coupled cells.
#Morris-Lecar dynamics with modified parameters.
#Notably: vc=-5, vd=10, phi=0.5  (originally: 2, 30, 0.04, respectively)
#Use weak gc (=1) for antiphase and strong gc (=2) for inphase.
#differential equations
dV1/dt=(I-gca*minf(V1)*(V1-Vca)-gk*w1*(V1-VK)-gl*(V1-Vl)+\
   gc*heav(t-ton)*(V2-V1))/c
dw1/dt = phi*(winf(V1)-w1)/tauw(V1)
dV2/dt = (I-gca*minf(V2)*(V2-Vca)-gk*w2*(V2-VK)-gl*(V2-Vl)+\
   gc*heav(t-ton)*(V1-V2))/c
dw2/dt = phi*(winf(V2)-w2)/tauw(V2)
init V1=-20,w1=.2
init V2=-30,w2=0
minf(v)=0.5*(1+tanh((v-va)/vb))
winf(v)=0.5*(1+tanh((v-vc)/vd))
tauw(v)=1/cosh((v-vc)/(2*vd))
#parameters
param vk=-84,vl=-60,vca=120
param i=-10,gk=8,gl=1,c=20
param va=-1.2,vb=18,gc=2,ton=100
param vc=-5,vd=10,phi=0.5,gca=8
#numerical parameters
@ total=400,dt=.25,xhi=400,ylo=-40,yhi=30
done
```

Figure 6.9. This file includes a delay using a Heaviside function. This file also demonstrates alternative notation for the equations. In particular, note that the `dv/dt` notation may be substituted for `v'`.

```
# simple model for coincidence detection.
# ML cell receives two just-subthreshold inputs that are identical and
# excitatory -  tdel represents timing difference between them.
# this ML model uses the standard ML (Type II) params from Chapt 2 of this
# book (same as Rinzel/Ermentrout in  Koch and Segev's book.
# Note: treat tdel as a variable (with ode: tdel'=0) so can do Poincare
# map for response tuning curve
isyn2=gsyne*s2*(Vsyne-v1)
isyn1=gsyne*s1*(Vsyne-v1)
#differential equations
v1' = (I-gca*minf(v1)*(v1-vca)-gk*w1*(v1-vk)-gl*(v1-vl)\
      +isyn2+isyn1)/c
w1' = phi*(winf(v1)-w1)/tauw(v1)
s2' = alphae*sinf(vr2(t))*(1-s2)-betae*s2
s1' = alphae*sinf(vr1(t))*(1-s1)-betae*s1
```

```
tdel'= 0
init v1=-60.9,w1=.0149,s2=0.,s1=0.,tdel=0
minf(v)=.5*(1+tanh((v-va)/vb))
winf(v)=.5*(1+tanh((v-vc)/vd))
tauw(v)=1/cosh((v-vc)/(2*vd))
sinf(v)=1/(1+exp(-(v-thetasyn)/ksyn))
# fix the start time of excitation from R2 and delay the
# start of excitation from R1  by tdel (if tdel<0, R1 input precedes
# R2 input)
vr2(t)=100*heav(t-t0)*heav(teon+t0-t)+vrest
vr1(t)=100*heav(t-t0-tdel)*heav(teon+t0+tdel-t)+vrest
param t0=100,teon=5,vrest=-60
param gsyne=1.5,vsyne=100
param vk=-84,vl=-60,vca=120
param i=0,gk=8,gl=2,c=20
param va=-1.2,vb=18,vc=2,vd=30,phi=.04,gca=4.
param thetasyn=20,ksyn=2,alphae=1,betae=0.3
#  aux quantities
aux isyn2x=isyn2
aux isyn1x=isyn1
@ total=400,bound=100000,nout=2,dt=.25,xhi=400,ylo=-100,yhi=100
done
```

Figure 7.6. This is the basic diffusion equation. It demonstrates the two–dimensional graphing capabilities of XPPAUT.

```
param D=1
param dx=1
c0'=D*(c1-c0)/(dx*dx)
c[1..39]'=D*(c[j+1]-2*c[j]+c[j-1])/(dx*dx)
c40'=D*(c39-c40)/(dx*dx)
init c[0..19]=0
init c20=0.5
init c[21..29]=1
init c30=0.5
init c[31..40]=0
```

Figure 7.8 This file provides a template for one-dimensional reaction–diffusion equations.

```
param dx=1
f(x)=x*(1-x)*(x-0.1)
V0'=(V1-V0)/(dx*dx)+f(V0)
V[1..39]'=(V[j+1]-2*V[j]+V[j-1])/(dx*dx)+f(V[j])
V40'=(V39-V40)/(dx*dx)+f(V40)
```

```
init V[0..30]=0
init V31=0.1
init V[32..40]=0.2
```

Figure 8.7. This file incorporates recovery, giving a traveling pulse with wave front and back.

```
# Calcium wave simulation
#parameters
param ip3=.7, a2=.2, Caer=1
param d1=0.1, d2=1, d3=0.2, d5=0.2
param v1=20, v2=0.004, v3=1.2, k3=0.15, tau=2
param deff=16,dx=10
# dx in units of um
# deff in units of um^2/sec
#the initial condition, equation and a function
Ca[0..4](0)=.1
Ca[5..100](0)=.01
w[0..100](0)=.8652
dCa0/dt=f(Ca0,w0)+deff*(Ca1-Ca0)/dx^2
dCa[1..99]/dt=f(Ca[j],w[j])+deff*(Ca[j-1]-2*Ca[j]+Ca[j+1])/dx^2
dCa100/dt = f(Ca100,w100)+deff*(Ca99-Ca100)/dx^2
dw[0..100]/dt=(winf(Ca[j])-w[j])/tau
f(Ca,w)=(v2+v1*(w*Ca/(Ca+d5))^3)*(Caer-Ca)-v3*Ca^2/(k3^2+Ca^2)
winf(Ca)=(d2*ip3/(ip3+d3))/(Ca+(d2*(ip3+d1)/(ip3+d3)))
#program end
@ total=200,trans=0,DT=.1,xlo=0,xhi=200,ylo=0,yhi=1
@ maxstore=1000000,bounds=10000
@ xplot=x,yplot=Ca0
done
# A forcing term can be added if you want
# capp = i1*(heav(mod(t,period))*heav(duty*period-mod(t,period)))
```

Figure 8.12A,B. The following file is interesting because although there are discrete release sites, one parameter set essentially results in continuous Ca^{2+} wave propagation seen in Figure 8.12A while another results in discrete release and saltatory propagation as seen in Figure 8.12B. To replicate the discrete release and saltatory propagation seen in Figure 8.12B, set `taur=0.01` and `total=0.5`. To replicate the continous wave of Figure 8.12A set `taur=1` and `total=2.0`. Use the 2DArray option in XPPAUT to view the results.

```
# Fire-diffuse-fire model simulation
# time in s
# space in um
# d in um^2/s
# parameters (here for continuous wave)
```

```
param d=30,dx=0.2,sigma=5,taur=1,taud=10000000,cth=0.1
#the initial condition, equation and a function
c[0..100](0)=0
s[0..20](0)=1.0
s[21..100](0)=0
#global 1 c0-cth {s0=1}
global 1 c10-cth {s10=1}
global 1 c20-cth {s20=1}
global 1 c30-cth {s30=1}
global 1 c40-cth {s40=1}
global 1 c50-cth {s50=1}
global 1 c60-cth {s60=1}
global 1 c70-cth {s70=1}
global 1 c80-cth {s80=1}
global 1 c90-cth {s90=1}
global 1 c100-cth {s100=1}
dc0/dt = sigma*heav(s0)/2/taur-c0/taud+d*(c1-c0)/dx^2
dc[1..99]/dt = sigma*heav(s[j])/taur-c[j]/taud+d*(c[j-1]-2*c[j]+c[j+1])/dx^2
dc100/dt = sigma*heav(s100)/2/taur-c100/taud+d*(c99-c100)/dx^2
ds[0..100]/dt = -heav(s[j])/taur
aux logc[0..100] = c[j]
#numerical parameters (total for continuous wave)
@ total=2,trans=0,dt=0.0001,xlo=0,xhi=2000,ylo=0,yhi=1
@ maxstore=1000000,bounds=10000
@ xplot=x,yplot=Ca10
done
```

Figure 9.10. This file incorporates an active delay into the equation.

```
# Discrete time lag oscillator
p p=4, b=.5, tau=2.5
dx/dt = 1/(1+delay(x,tau)^p) - b*x
x(0)=1.1
@ delay=10
done
```

Figure 11.3. This file shows the basic two-state stochastic model.

```
# Example two state channel simulation
# parameters
params kp=1.5, km=0.5, tau=1000
#initial condition
po(0)=0.5
#stochastic variable
markov n 2
```

```
{0}{kp}
{km}{0}
#differential equation
po'=-(po-n)/tau
aux n=n
#numerical parameters
@ total=100,trans=0,DT=.001,xlo=0,xhi=100,ylo=-0.1,yhi=1.1
@ maxstore=1000000,bounds=10000
@ xplot=t,yplot=n
#@ njmp=100
done
```

Figure 11.9. This files introduces Wiener variables.

```
#weiner variables
wiener w[0..99]
#differential equations
x[0..99]'=w[j]
mean = sum(0,99)of(shift(x0,i'))/100
aux mean = mean
aux var = sum(0,99)of((shift(x0,i')-mean)^2)/100
aux w0 = w0
#numerical parameters
@ total=1000,trans=0,DT=0.1,xlo=0,xhi=1000,ylo=-100,yhi=100
@ maxstore=1000000,bounds=10000
@ xplot=t,yplot=x
@ njmp=1
done
```

Figure 11.11A. The next two files are used to explore the differences between twenty states and two states in a Markov process. Notice that the rows and columns of the transition probability matrix follow the XPPAUT convention and are transposed relative to (11.5). Diagonal entries are not used by XPPAUT.

```
#parameters
params kp=0.5, km=0.5
params c=2, gl=0.5, gch=1.0, vl=-70, vch=20
#initial condition
v(0)=0
#stochastic variable
markov n 2
{0}{kp}
{km}{0}
#differential equation
v'=(-gl*(v-vl)-gch*n*(v-vch))/c
```

```
aux n=n
#numerical parameters
@ total=100,trans=0 ,DT=.001,xlo=0,xhi=100,ylo=-80,yhi=0
@ maxstore=1000000,bounds=10000
done
```

Figure 11.11C. This is the twenty–state Markov process. Again, the transition probability matrix follows the XPPAUT convention and is transposed relative to (11.13).

```
#parameters
params kp=0.5, km=0.5
params c=2, gl=0.5, gch=1.0, vl=-70, vch=20
#initial condition
v(0)=0
#stochastic variable
markov n 21
{0}{20*kp}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}
{km}{0}{19*kp}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}
{0}{2*km}{0}{18*kp}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}
{0}{0}{3*km}{0}{17*kp}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}
{0}{0}{0}{4*km}{0}{16*kp}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}
{0}{0}{0}{0}{5*km}{0}{15*kp}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}
{0}{0}{0}{0}{0}{6*km}{0}{14*kp}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}
{0}{0}{0}{0}{0}{0}{7*km}{0}{13*kp}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}
{0}{0}{0}{0}{0}{0}{0}{8*km}{0}{12*kp}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}
{0}{0}{0}{0}{0}{0}{0}{0}{9*km}{0}{11*kp}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}
{0}{0}{0}{0}{0}{0}{0}{0}{0}{10*km}{0}{10*kp}{0}{0}{0}{0}{0}{0}{0}{0}{0}
{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{11*km}{0}{9*kp}{0}{0}{0}{0}{0}{0}{0}{0}
{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{12*km}{0}{8*kp}{0}{0}{0}{0}{0}{0}{0}
{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{13*km}{0}{7*kp}{0}{0}{0}{0}{0}{0}
{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{14*km}{0}{6*kp}{0}{0}{0}{0}{0}
{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{15*km}{0}{5*kp}{0}{0}{0}{0}
{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{16*km}{0}{4*kp}{0}{0}{0}
{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{17*km}{0}{3*kp}{0}{0}
{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{18*km}{0}{2*kp}{0}
{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{19*km}{0}{kp}
{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{20*km}{0}
#differential equation
v'=(-gl*(v-vl)-gch*n/20*(v-vch))/c
aux n=n
#numerical parameters
@ total=100,trans=0 ,DT=.001,xlo=0,xhi=100,ylo=-80,yhi=0
@ maxstore=1000000,bounds=10000
```

```
done
```

Figure 11.14. This file incorporates stochastic dynamics into the differential equations for the Morris–Lecar model.

```
# Deterministic Morris-Lecar model follows Rinzel and Ermentrout's
# chapter in Koch & Segev.  Stochastic ion channel dynamics added
# using Langevin formulation.
# For stochastic excitability, oscillations, and bistability use
# iapp = 10, 12, 10 and v3 = 10, 10, 15
#initial conditions
v(0)=-40
w(0)=1
#weiner variable
wiener b
#parameters
params n=100
params v1=-1,v2=15,v3=10,v4=14.5,gca=1.33,phi=.333
params vk=-70,vl=-50,iapp=10,gk=2.0,gl=.5,om=1
minf(v)=.5*(1+tanh((v-v1)/v2))
# The 0.05 is a modification needed to lift w nullcline
# so that stochastic excitability can be realized
ninf(v)=.5*(1+tanh((v-v3)/v4))+0.05
lamn(v)= phi*cosh((v-v3)/(2*v4))
ica=gca*minf(v)*(v-100)
#differential equations
v'=  (iapp+gl*(vl-v)+gk*w*(vk-v)-ica)*om
w'= (lamn(v)*(ninf(v)-w))*om+sqrt(lamn(v)*((1-2*ninf(v))*w+ninf(v))/n)*b
aux I_ca=ica
#numerical parameters
@ total=500,trans=0,DT=.01,xlo=0,xhi=500,ylo=-60,yhi=50
@ maxstore=1000000,bounds=10000
@ xplot=t,yplot=v
done
```

References

Alberts, B., Bray, D., Lewis, J., Raff, M., Roberts, K. and Watson, J. (1994). *Molecular Biology of the Cell*, 3rd edition, Garland Publishing, New York.

Allbritton, N., Meyer, T. and Stryer, L. (1992). Range of messenger action of calcium ion and inositol 1,4,5-trisphosphate, *Science* **258**, 1812.

Anderson, C. and Stevens, C. (1973). Voltage clamp analysis of acetylcholine produced end-plate current fluctuations at frog neuromuscular junction, *J. Physiol. (Lond.)* **235**(3), 655–691.

Asher, U. and Petzold, L. (1998). *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, 1st edition, SIAM.

Atri, A., Amundson, J., Clapham, D. and Sneyd, J. (1993). A single pool model for intracellular calcium oscillations and waves in the *Xenopus laevis* oocyte, *Biophys. J.* **65**, 1727–1739.

Atwater, I., Dawson, C., Scott, A., Eddlestone, G. and Rojas, E. (1980). The nature of the oscillatory behavior in electrical activity for pancreatic beta cell., *Horm. Metab. Res. Suppl. 1980* **Suppl. 10**, 100–107.

Bell, G., Burant, C., Takeda, J. and Gould, G. (1993). Structure and function of mammalian facilitative sugar transporters, *J. Biol. Chem.* **268**, 19161–19164.

Berg, H. (1993). *Random Walks in Biology*, expanded edition, Princeton University Press, Princeton, N.J.

Berg, H. (2000). Constraints on models for the flagellar rotary motor, *Phil. Trans. R. Soc. Lond. B* **355**, 491–501.

Berridge, M. (1997). Elementary and global aspects of calcium signalling, *J. Physiol. (Lond.)* **499**, 291.

Berridge, M. (1998). Neuronal calcium signaling, *Neuron* **21**, 13.

Berridge, M. and Rapp, P. (1979). A comparative survey of the function, mechanism, and control of cellular oscillators, *J. Exp. Biol.* **81**, 217–279.

Berry, R. (1993). Torque and switching in the bacterial flagellar motor. An electrostatic model, *Biophys. J.* **64**(4), 961–973.

Bertram, R., Smolen, P., Sherman, A., Mears, D., Atwater, I., Martin, F. and Soria, B. (1995). A role for calcium release-activated current (CRAC) in cholinergic modulation of electrical activity in pancreatic β-cells, *Biophys. J.* **68**, 2323–2332.

Bezprozvanny, I., Watras, J. and Ehrlich, B. (1991). Bell-shaped calcium response curves of Ins(1,4,5)P_3- and calcium-gated channels from endoplasmic reticulum of cerebellum, *Nature* **351**, 751–754.

Billing, G. D. and Mikkelsen, K. V. (1996). *Introduction to Molecular Dynamics and Chemical Kinetics*, Wiley, New York.

Bliss, R., Painter, P. and Marr, A. (1982). Role of feedback inhibition in stabilizing the classical operon, *J. Theor. Biol.* **97**, 177–193.

Bohnensack, R. (1982). The role of the adenine nucleotide translocator in oxidative phosphorylation. A theoretical investigation on the basis of a comprehensive rate law of the translocator, *J. Bioenerg. Biomembr.* **14**, 45–61.

Boissonade, J. and De Kepper, P. (1980). Transitions from bistability to limit cycle oscillations. Theoretical analysis and experimental evidence in an open chemical system, *J. Phys. Chem.* **84**, 501–506.

Borisuk, M. and Tyson, J. (1998). Bifurcation analysis of a model of mitotic control in frog eggs, *J. Theor. Biol.* **195**, 69–85.

Carslaw, H. and Jaeger, J. (1959). *Conduction of Heat in Solids*, 2nd edition, Clarendon Press, Oxford.

Chance, B., Pye, P., Ghosh, A. and Hess, B. (1973). *Biological and Biochemical Oscillators*, Academic Press, New York.

Chauwin, J.-F., Oster, G. and Glick, B. (1998). Protein import into mitochondria: A comparison of the Brownian ratchet and power stroke models, *Biophys. J.* **74**(4), 1732–1743.

Chay, T. (1997). Effects of extracellular calcium on electrical bursting and intracellular and luminal calcium oscillations in insulin secreting pancreatic β-cells, *Biophys. J.* **73**, 1673–1688.

Chay, T. and Keizer, J. (1983). Minimal model for membrane oscillations in the pancreatic Beta cell, *Biophys. J.* **42**, 181–190.

Chen, K., Csikasz-Nagy, A., Gyorffy, B., Val, J., Novak, B. and Tyson, J. J. (2000). Kinetic analysis of a molecular model of the budding yeast cell cycle, *Mol. Biol. Cell* **11**, 369–391.

Cheng, H., Lederer, W. and Cannell, M. (1993). Calcium sparks: elementary events underlying excitation-contraction coupling in heart muscle, *Science* **262**, 740.

Cheng, H., Lederer, W. and Cannell, M. (1996). Calcium sparks and $[Ca^{2+}]_i$ waves in cardiac myocytes, *Am. J. Physiol.* **270**, C148–59.

Chow, C. and White, J. (1996). Spontaneous action potentials due to channel fluctuations, *Biophys. J.* **71**, 3013–21.

Cole, K. (1968). *A Quantitative Description of Membrane Current and Its Application to Conductance and Excitation in Nerve.*, University of California Press, Berkeley.

Cole, N., Smith, C., Sciaky, N., Terasaki, M. and Edidin, M. (1996). Diffusional mobility of golgi proteins in membranes of living cells, *Science* **273**, 797–801.

Cooper, G. M. (1997). *The Cell: A Molecular Approach*, ASM Press, Washington DC.

Crank, J. (1975). *The Mathematics of Diffusion*, 2nd edition, Clarendon Press, Oxford.

Cuthbertson, K. and Chay, T. (1991). Modelling receptor-controlled intracellular calcium oscillators, *Cell Calcium* **12**, 97–109.

Dayan, P. and Abbott, L. (2001). *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*, MIT Press, Cambridge, MA.

delCastillo, J. and Moore, J. (1959). On increasing the velocity of a nerve impulse, *J. Physiol. (London)* **148**, 665–670.

Derenyi, I. and Vicsek, T. (1996). The kinesin walk: a dynamic model with elastically coupled heads, *Proc. Natl. Acad. Sci.* **93**, 6775–6779.

Destexhe, A., Mainen, Z. and Sejnowski, T. (1998). Kinetic models of synaptic transmission, *in Methods in Neuronal Modeling*, C. Koch and I. Segev (eds.), The MIT Press, Cambridge.

DeYoung, G. and Keizer, J. (1992). A single-pool inositol 1,4,5-trisphosphate-receptor-based model for agonist-stimulated oscillations in Ca^{2+} concentration, *Proc. Natl. Acad. Sci. USA* **89**, 9895–9899.

Dimroth, P., Wang, H., Grabe, M. and Oster, G. (1999). Energy transduction in the sodium F-ATPase of *Propionigenium modestum*, *Proc. Natl. Acad. Sci. USA* **96**(9), 4924–4929.

Doering, C. (1990). Modeling complex systems: Stochastic processes, stochastic differential equations, and Fokker-Planck equations, *in 1990 Lectures in Complex Systems*, L. Nadel and D. Stein (eds.), Vol. III of *Sante Fe Institute Studies in the Sciences of Complexity*, Addison-Wesley, Redwood City, CA, pp. 3–51.

Doering, C. (1995). Randomly rattled ratchets, *Nuovo Cimento D* **17**, 685–97.

Doering, C. (1998). Stochastic ratchets, *Physica A* **254**, 1–6.

Dogterom, M. and Yurke, B. (1997). Measurement of the force-velocity relationship for growing microtubules, *Science* **278**, 856–860.

Doi, M. and Edwards, S. (1986). *The Theory of Polymer Dynamics*, Oxford University Press, New York.

Dolmetsch, R. and Lewis, R. (1994). Signaling between intracellular Ca^{2+} stores and depletion-activated Ca^{2+} channels generates $[Ca^{2+}]_i$ oscillations in T lymphocytes, *J Gen Physiol.* **103**(3), 365–388.

Dunlap, J. (1999). Molecular bases for circadian clocks, *Cell* **96**, 271–290.

Dupont, G. and Goldbeter, A. (1994). Properties of intracellular Ca^{2+} waves generated by a model based on Ca^{2+}-induced Ca^{2+} release, *Biophys. J.* **67(6)**, 2191–204.

Eckert, R. and Tillotson, D. (1981). Calcium-mediated inactivation of the calcium conductance in caesium-loaded giant neurons of *Aplysia*, *J. Physiol. (Lond)* **314**, 265–280.

Edelstein-Keshet, L. (1988). *Mathematical Models in Biology*, 1st edition, McGraw-Hill, New York.

Edwards, Jr., L. (1988). *Cellular and Molecular Bases of Biological Clocks*, 1st edition, Springer, New York.

Elston, T. (2000). Models of post-translational protein translocation, *Biophys. J.* **79**(5), 2235–2251.

Elston, T. and Doering, C. (1995). Numerical and analytical studies of nonequilibrium fluctuation-induced transport processes, *J. Stat. Phys.* **83**, 359–383.

Elston, T. and Oster, G. (1997). Protein turbines I: The bacterial flagellar motor, *Biophys. J.* **73**(2), 703–721.

Elston, T., Wang, H. and Oster, G. (1998). Energy transduction in ATP synthase, *Nature* **391**, 510–514.

Ermentrout, B. (1998). Neural networks as spatio-temporal pattern-forming systems, *Rep. Prog. Phys.* **61**, 353–430.

Ermentrout, B. (2002). *Simulating, Analyzing, and Animating Dynamical Systems: A Guide to XPPAUT for Researchers and Students*, 1st edition, SIAM, New York.

Feynman, R., Leighton, R. and Sands, M. (1963). *The Feynman Lectures on Physics*, Addison-Wesley, Reading, MA. Ratchet and Pawl: Vol. 1, Chap 46.

Finch, E., Turner, T. and Goldin, S. (1991). Calcium as a coagonist of inositol 1,4,5-trisphosphate-induced calcium release, *Science* **252**, 443–446.

FitzHugh, R. (1961). Impulses and physiological states in models of nerve membrane, *Biophys J.* **1**, 445–466.

Fox, R. (1997). Stochastic versions of the Hodgkin–Huxley equations, *Biophys. J.* **72**, 2068–2074.

Fox, R. and Choi, M. (2001). Rectified Brownian motion and kinesin motion along microtubules, *Phys. Rev. E*.

Friel, D. (1995). $[Ca^{2+}]_i$ oscillations in sympathetic neurons: an experimental test of a theoretical model, *Biophys. J.* **68**(5), 1752–1766.

Friel, D. and Tsien, R. W. (1992). Phase-dependent contributions from Ca^{2+} entry and Ca^{2+} release to caffeine-induced Ca^{2+} oscillations in bullfrog sympathetic neurons, *Neuron* **8**, 1109–1125.

Fygenson, D., Marko, J. and Libchaber, A. (1997). Mechanics of microtubule-based membrane extension, *Phys. Rev. Lett.* **79**(22), 4497–500.
Gantmacher, F. (1959). *Theory of Matrices, Volume II*, Chelsea Publishing, New York.
Gardiner, C. W. (1997). *Handbook of Stochastic Methods*, 3rd edition, Springer Verlag, New York.
Gillespie, D. (1977). Exact stochastic simulation of coupled chemical reactions, *J Phys. Chem.* **81**, 2340–2361.
Girard, S., Luckhoff, A., Lechleiter, J., Sneyd, J. and Clapham, D. (1992). Two-dimensional model of calcium waves reproduces the patterns observed in *Xenopus* oocytes, *Biophys. J.* **61(2)**, 509–17.
Goldbeter, A. (1995). A model for circadian oscillations in the *Drosophila* period protein (PER), *Proc. R. Soc. Lond. B* **261**, 319–324.
Goldbeter, A. (1996). *Biochemical Oscillations and Cellular Rhythms*, Cambridge Univ. Press, Cambridge, UK.
Goldbeter, A. and Koshland, D. (1981). An amplified sensitivity arising from covalent modification in biological systems, *Proc. Natl. Acad. Sci. U.S.A.* **78**, 6840–6844.
Goldbeter, A., Dupont, G. and Berridge, M. (1990). Minimal model for signal-induced Ca^{2+} oscillations and for their frequency encoding through protein phosphorylation, *Proc. Natl. Acad. Sci. USA* **87**, 1461–1465.
Goodwin, B. (1965). Oscillatory behavior in enzymatic control processes, *Adv. Enz. Regul.* **3**, 425–438.
Goodwin, B. (1966). An entrainment model for timed enzyme synthesis in bacteria, *Nature* **209**, 479–481.
Griffith, J. (1968a). Mathematics of cellular control processes. I. Negative feedback to one gene, *J. Theor. Biol.* **20**, 202–208.
Griffith, J. (1968b). Mathematics of cellular control processes. II. Positive feedback to one gene, *J. Theor. Biol.* **20**, 209–216.
Grimes, A. (1980). *Human Red Cell Metabolism*, 1st edition, Blackwell Sciences, New York.
Grodsky, G. (1972). A threshold distribution hypothesis for packet storage of insulin and its mathematical modeling, *J. Clin. Invest.* **51**, 2047–2059.
Grynkiewicz, G., Poenie, M. and Tsien, R. Y. (1985). A new generation of Ca^{2+} indicators with greatly improved fluorescence properties, *J. Biol. Chem.* **260**(6), 3440–50.
Guckenheimer, J. (1986). Multiple bifurcation problems for chemical reactors, *Physica D* **20**, 1–20.
Györke, S. and Fill, M. (1993). Ryanodine receptor adaptation: control mechanism of Ca^{2+}-induced Ca^{2+} release in heart, *Science* **260**, 807–809.
Hanggi, P., Talkner, P. and Borkovec, M. (1990). Reaction-rate theory: 50 years after Kramers, *Rev. Mod. Phys.* **62**, 254–341.
Hansel, D. and Sompolinsky, H. (1998). Modeling feature selectivity in local cortical circuits, *in Methods in Neuronal Modeling*, C. Koch and I. Segev (eds.), The MIT Press, Cambridge.
Happel, J. and Brenner, H. (l986). *Low Reynolds Number Hydrodynamics*, Vol. 1 of *Mechanics of fluids and transport processes*, Nijhoff, The Hague.
Higgins, J. (1967). The theory of oscillating reactions, *Ind. Eng. Chem.* **59**, 18–62.
Hill, T. (1977). *Free Energy Transduction in Biology*, Academic Press, New York.
Hille, B. (2001). *Ionic Channels of Excitable Membranes*, 3rd edition, Sinauer Associates, Inc., Sunderland.
Hodgkin, A. and Huxley, A. (1939). Action potentials recorded from inside a nerve fibre, *J. Physiol (Lond.)* **144**, 710–711.
Hodgkin, A. and Huxley, A. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve., *J. Physiol (Lond.)* **117**, 500–544.

Honda, M., Takiguchi, K., Ishikawa, S. and Hotani, H. (1999). Morphogenesis of liposomes encapsulating actin depends on the type of actin-crosslinking, *J. Mol. Biol.* **287**, 293–300.
Howard, J. (2001). *Mechanics of Motor Proteins and the Cytoskeleton*, 1st edition, Sinauer Associates, Sunderland, MA.
Huxley, A. (1957). Muscle structure and theories of contraction, *Prog. Biophys. Biophys. Chem.* **7**, 255–318.
Huxley, A. and Simmons, R. (1971). Proposed mechanism of force generation in striated muscle, *Nature* **233**, 533–538.
Iino, M. (1990). Biphasic Ca^{2+}-dependence of inositiol 1,4,5-trisphosphate-induced Ca^{2+} release in smooth muscle cells of the guinea pig *Taenia caeci* gonadotrophs, *J. Gen. Physiol.* **95**, 1103–1122.
Israelachvili, J. (1992). *Intermolecular and Surface Forces*, 2nd edition, Academic Press, New York.
Jafri, M. and Keizer, J. (1994). Diffusion of inositol 1,4,5-trisphosphate but not Ca^{2+} is necessary for a class of inositol 1,4,5-trisphosphate-induced Ca^{2+} waves, *Proc. Natl. Acad. Sci. USA* **91**, 9485.
Jafri, M. and Keizer, J. (1997). Agonist-induced calcium waves in oscillatory cells: a biological example of Burgers' equation, *Bull Math Biol.* **59**(6), 1125–44.
Johnston, D. and Wu, S. (1995). *Foundations of Cellular Neurophysiology*, 1st edition, MIT Press, Cambridge.
Julicher, F. and Bruinsma, R. (1998). Motion of RNA polymerase along DNA: A stochastic model, *Biophys. J.* **74**(3), 1169–1185.
Kandel, E., Schwartz, J. and Jessell, T. (eds.) (2000). *Principles of Neural Science*, 4th edition, McGraw-Hill, New York.
Kaplan, D. and Glass, L. (1995). *Understanding Nonlinear Dynamics*, 1st edition, Springer Verlag, New York.
Keener, J. (1999). *Principles of Applied Mathematics*, Perseus Books, New York.
Keener, J. (2000). Propagation of waves in an excitable medium with discrete release sites, *SIAM J. Appl. Math* **61**, 317–334.
Keener, J. and Sneyd, J. (1998). *Mathematical Physiology*, 1st edition, Springer, New York.
Keizer, J. (1987). *Statistical Thermodynamics of Nonequilibrium Processes*, Springer-Verlag, New York.
Keizer, J. and Levine, L. (1996). Ryanodine receptor adaptation and Ca^{2+}-induced Ca^{2+} release-dependent Ca^{2+} oscillations, *Biophys. J.* **71**, 3477–3487.
Keizer, J. and Magnus, G. (1989). ATP-sensitive potassium channel and bursting in the pancreatic beta-cell, *Biophys. J.* **56**, 229–242.
Keizer, J., Li, Y.-X., Stojilković, S. and Rinzel, J. (1995). $InsP_3$-induced Ca^{2+} excitability of the endoplasmic reticulum, *Molecular Biology of the Cell* **6**, 945–951.
Keizer, J., Smith, G., Ponce-Dawson, S. and Pearson, J. (1998). Saltatory propagation of Ca^{2+} waves by Ca^{2+} sparks, *Biophys. J.* **75**, 595.
Keller, A. (1995). Model genetic circuits encoding autoregulating transcription factors, *J. Theor. Biol.* **172**, 169–85.
Keller, D. and Bustamante, C. (2000). The mechanochemistry of molecular motors, *Biophys. J.* **78**, 541–556.
Kirk, J., Orr, J. and Hope, C. (1968). A mathematical analysis of red blood cell and bone marrow stem cell control mechanisms, *Brit. J. Haematology* **15**, 35–46.
Koch, C. (1999). *Biophysics of Computation : Information Processing in Single Neurons*, Oxford University Press, New York.
Koch, C. and Segev, I. (2000). The role of single neurons in information processing, *Nat. Neurosci* **3**, 1171–1177.

Koch, C. and Segev, I. (eds.) (1998). *Methods in Neuronal Modeling*, 2nd edition, MIT Press, Cambridge.

Kohn, K. (1999). Molecular interaction map of the mammalian cell cycle control and DNA repair systems, *Mol. Biol. Cell* **10**, 2703–2734.

Konopka, R. and Benzer, S. (1971). Clock mutants in *Drosophila melanogaster*, *Proc. Natl. Acad. Sci. USA* **68**, 2112–2116.

Kuba, K. and Nishi, S. (1976). Rhythmic hyperpolarizations and depolarization of sympathetic ganglion cells induced by caffeine, *J. Neurophysiol.* **39**, 547–563.

Kuba, K. and Takeshita, S. (1981). Simulation of intracellular Ca^{2+} oscillation in a sympathetic neuron, *J. Theor. Biol.* **93**, 1009–1031.

Kukuljan, M., Rojas, E., Catt, K. and Stojilković, S. (1994). Membrane potential regulates inositol 1,4,5-trisphosphate-controlled cytoplasmic Ca^{2+} oscillations in pituitary gonadotrophs, *J. Biol. Chem.* **269**, 4860–4865.

Kuznetsov, Y. (1998). *Elements of Applied Bifurcation Theory*, 2nd edition, Springer-Verlag, New York.

Landau, L., Lifshitz, E. and Pitaevskii, L. (1980). *Statistical Physics*, 3rd edition, Pergamon Press, Oxford.

Läuger, P. (1990). Microscopic models of the bacterial flagellar motor, *Comments Theoret. Biol.* **2**, 99–123.

Läuger, P. (1991). F0F1-ATPases, *in Electrogenic Ion Pumps*, L. Nadel and D. Stein (eds.), Addison-Wesley, pp. 252–269.

Lawrence, P. (1992). *The Making of a Fly: The Genetics of Animal Design*, Blackwell Science, Oxford.

Lechleiter, J. and Clapham, D. (1992). Molecular mechanisms of intracellular calcium excitability in *X. laevis* oocytes, *Cell.* **69**, 283–294.

Leinhard, G., Slot, J., James, D. and Mueckler, M. (1992). How cells absorb glucose, *Scientific American* **January**, 89.

Leloup, J. and Goldbeter, A. (1998). A model for circadian rhythms in *Drosophila* incorporating the formation of a complex between the PER and TIM proteins, *J. Biol. Rhythms* **13**, 70–87.

Leloup, J. and Goldbeter, A. (1999). Limit cycle models for circadian rythms based on transcriptional regulation in *Drosophila* and *neurospora*, *J. Biol. Rhythms* **14**, 433–448.

Lema, M., Golombek, D. and Echave, J. (2000). Delay model of the circadian pacemaker, *J. Theor. Biol.* **204**, 565–573.

Li, Y.-X. and Rinzel, J. (1994). Equations for $InsP_3$ receptor-mediated Ca^{2+} oscillations derived from a detailed kinetic model: A Hodgkin–Huxley like formalism, *J. Theor. Biol.* **166**, 461–473.

Li, Y.-X., Keizer, J., Stojilković, S. and Rinzel, J. (1995b). Ca^{2+} excitability of the ER membrane: an explanation for IP_3-induced Ca^{2+} oscillations, *Am. J. Phys.* **269**, C1079–C1092.

Li, Y.-X., Rinzel, J., Vergara, L. and Stojilković, S. (1995a). Spontaneous electrical and calcium oscillations in unstimulated pituitary gonadotrophs, *Biophys. J.* **69**, 785–795.

Li, Y.-X., Stojilković, S., Keizer, J. and Rinzel, J. (1997). Sensing and refilling calcium stores in an excitable cell, *Biophys. J.* **72**, 1080–1091.

Lindenberg, K. and Seshadri, V. (1979). Analytic theory of extrema. I. Asymptotic theory for Fokker-Planck processes, *J. Chem. Phys.* **71**(10), 4075–4084.

Logan, D. (1997). *Applied Mathematics*, John Wiley and Sons, New York.

Mackey, M. (1996). Periodic hemolytic anemia, *in Case Studies in Mathematical Modeling: Ecology, Physiology and Cell Biology*, H. Othmer, F. Adler, J. Dallon and M. Lewis (eds.), Prentice Hall, Upper Saddle River, NJ, pp. 149–178.

Mahadevan, L. and Matsudaira, P. (2000). Motility powered by supramolecular springs and ratchets, *Science* **288**, 95–99.

Maki, L. and Keizer, J. (1995). Mathematical analysis of a proposed mechanism for oscillatory insulin secretion in perifused HIT-15 cells, *Bull. Math. Biology* **57**, 569–591.

McCormick, D. (1999). Membrane potential and action potential, *in Fundamental Neuroscience*, M. J. Zigmond, F. E. Bloom, S. C. Landis, J. L. Roberts and L. R. Squire (eds.), Academic Press, New York.

McLaughlin, D., Shapley, R., Shelley, M. and Wielaard, D. (2000). A neuronal network model of macaque primary visual cortex (v1): orientation selectivity and dynamics in the input layer 4calpha, *Proc. Natl. Acad. Sci. USA.* **97**(14), 8087–8092.

McMahon, T. (1984). *Muscles, Reflexes, and Locomotion*, Princeton University Press, Princeton.

Meinhardt, H. (1998). *The Algorithmic Beauty of Sea Shells*, Springer Verlag, New York.

Meyer, T. and Stryer, L. (1988). Molecular model for receptor-stimulated calcium spiking, *Proc. Natl. Acad. Sci. USA* **85**, 5051–5055.

Mogilner, A. and Oster, G. (1996). Cell motility driven by actin polymerization, *Biophys. J.* **71**(6), 3030–3045.

Mogilner, A. and Oster, G. (1999). The polymerization ratchet model explains the force-velocity relation for growing microtubules, *Eur. J. Biophys.* **28**(3), 235–242.

Morris, C. and Lecar, H. (1981). Voltage oscillations in the barnacle giant muscle, *Biophys J.* **35**, 193–213.

Murray, A. and Hunt, T. (1993). *The Cell Cycle. An Introduction*, W.H. Freeman and Co., New York.

Murray, J. (1981). A pre-pattern formation mechanism for animal coat markings, *J. Theor. Biol.* **88**, 161–99.

Murray, J. (1989). *Mathematical Biology*, 1st edition, Springer-Verlag, New York.

Nagumo, J., Arimoto, S. and Yoshizawa, S. (1962). An active pulse transmission line simulating nerve axon, *Proc. IRE.* **50**, 2061–2070.

Naraghi, M. and Neher, E. (1997). Linearized buffered Ca^{2+} diffusion in microdomains and its implications for calculation of $[Ca^{2+}]$ at the mouth of a calcium channel, *J. Neurosci.* **17**, 6961.

Naray-Szabo, G. and Warshel, A. (1997). *Computational Approaches to Biochemical Reactivity*, Vol. 19 of *Understanding chemical reactivity*, Kluwer Academic, Boston.

Nasmyth, K. (1996). At the heart of the budding yeast cell cycle, *Trends Genet.* **12**, 405–412.

Neher, E. (1986). Concentration profiles of intracellular Ca^{2+} in the presence of diffusible chelator, *Exp. Brain Res.* **14**, 80.

Neher, E. (1998). Usefulness and limitations of linear approximations to the understanding of Ca^{2+} signals, *Cell Calcium* **24**, 345.

Nohmi, M., Hua, S.-Y. and Kuba, K. (1992). Basal Ca^{2+} and the oscillation of Ca^{2+} in caffeine-treated bullfrog sympathetic neurones, *J. Physiol. (Lond.)* **450**, 513–528.

Nuccitelli, R., Yim, D. L. and Smart, T. (1993). The sperm-induced Ca^{2+} wave following fertilization of the *Xenopus* egg requires the production of Ins(1,4,5)P3, *Dev. Biol.* **158**(1), 200–12.

Nykamp, D. and Tranchina, D. (2000). A population density approach that facilitates large scale modeling of neural networks: Analysis and an application to tuning, *J. Comput. Neurosci.* **8**(1), 19–50.

O'Donovan, M., Wenner, P., Chub, N., Tabak, J. and Rinzel, J. (1998). Mechanisms of spontaneous activity in the developing spinal cord and their relevance to locomotion, *Ann. N.Y. Acad. Sci. USA* **860**, 130–141.

Omurtag, A., Knight, B. and Sirovich, L. (2000). On the simulation of large populations of neurons, *J. Comput. Neurosci.* **8**(1), 51–64.

Oosawa, F. and Hayashi, S. (1986). The loose coupling mechanism in molecular machines of living cells, *Adv Biophys* **22**, 151–183.

Oster, G. and Wang, H. (2000a). Reverse engineering a protein: The mechanochemistry of ATP synthase, *Bioc. Biophys. Acta (Bioenergetics)* **1458**(1-2), 482–510.

Oster, G. and Wang, H. (2000b). Why is the efficiency of the F1 ATPase so high?, *J. Bioenerg. Biomembr.* **32**, 459–469.

Oster, G., Perelson, A. and Tilney, L. (1982). A mechanical model for acrosomal extension in *Thyone*, *J. Math. Biol* **15**, 259–65.

Othmer, H. and Tang, Y. (1993). Oscillations and waves in a model of $InsP_3$-controlled calcium dynamics, *in Experimental and Theoretical Advances in Biological Pattern Formation*, H. G. Othmer, P. Maini and J. Murray (eds.), Plenum Press, New York, pp. 277–299.

Pape, P., Jong, D. and Chandler, W. (1998). Effects of partial sarcoplasmic reticulum calcium depletion on calcium release in frog cut muscle fibers equilibrated with 20 mM EGTA, *J. Gen. Physiol.* **112**(3), 263–95.

Parent, L., Supplisson, S., Loo, D. and Wright, E. (1992a). Electrogenic properties of the cloned Na^+/glucose cotransporter: I. Voltage clamp studies, *J. Membrane Biol.* **125**, 49–62.

Parent, L., Supplisson, S., Loo, D. and Wright, E. (1992b). Electrogenic properties of the cloned Na^+/glucose cotransporter: II. A transport model under nonrapid equilibrium conditions, *J. Membrane Biol.* **125**, 63–79.

Parker, I. and Ivorra, I. (1990). Inhibition by Ca^{2+} of inositol trisphosphate-mediated Ca^{2+} liberation - a possible mechanism for oscillatory release of Ca^{2+}, *Proc. Natl. Acad. Sci. USA* **87**(1), 260–264.

Parker, I., Choi, J. and Yao, Y. (1996). Elementary events of InsP3-induced Ca^{2+} liberation in *Xenopus* oocytes: hot spots, puffs and blips, *Cell Calcium* **20**, 105.

Pearson, J. (1993). Complex patterns in a simple system, *Science* **261**, 189–192.

Perez Velasquez, J. and Carlen, P. (2000). Gap junctions, synchrony and seizures, *Trends Neurosci.* **23**(2), 68–74.

Pernarowski, M., Miura, R. and Kevorkian, J. (1992). Perturbation techniques for models of bursting electrical activity in the pancreatic Beta cells, *SIAM J. Appl. Math.* **52**, 1627–1650.

Peskin, C. and Oster, G. (1995). Coordinated hydrolysis explains the mechanical behavior of kinesin, *Biophys. J.* **68**(4), 202s–210s.

Peskin, C., Odell, G. and Oster, G. (1993). Cellular motions and thermal fluctuations: the Brownian ratchet, *Biophys. J.* **65**, 316–324.

Pietrobon, D. and Caplan, S. (1985). Flow-force relationships for a six-state proton pump model: intrinsic uncoupling, kinetic equivalence of input and output forces, and domain of approximate linearity, *Biochemistry* **24**, 5764–5778.

Pinto, D., Brumberg, J., Simons, D. and Ermentrout, G. (1996). A quantitative population model of whisker barrels: re-examining the Wilson-Cowan equations, *J. Comput. Neurosci.* **3**(3), 5247–264.

Ponce-Dawson, S., Keizer, J. and Pearson, J. (1999). Fire-diffuse-fire model of dynamics of intracellular calcium waves, *Proc. Natl. Acad. Sci. USA* **96**, 6060.

Porte, Jr., D. (1990). Banting lecture 1990. Beta-cells in type II diabetes mellitus., *Diabetes* **40**(2), 166–180.

Pratusevich, V. and Balke, C. (1996). Factors shaping the confocal image of the calcium spark in cardiac muscle cells, *Biophys. J.* **71**(6), 2942–57.

Pye, E. (1971). Periodicities in intermediary metabolism, *in Biochronometry*, M. Menaker (ed.), National Acad. Sci. Press, Washington, DC, pp. 623–636.

Quon, M. and Campfield, L. (1991a). A mathematical model and computer simulation study of insulin sensitive glucose transporter regulation, *J Theor. Biol.* **150**, 93–107.

Quon, M. and Campfield, L. (1991b). A mathematical model and computer simulation study of insulin receptor regulation, *J Theor. Biol.* **150**, 59–72.

Rapp, P. (1979). An atlas of cellular oscillators, *J. Exp. Biol* **81**, 281–306.

Reif, F. (1965). *Fundamentals of Statistical and Thermal Physics*, McGraw Hill, New York.
Rieke, F., Warland, D., de Ruyter van Steveninck, R. and Bialek, W. (1997). *Spikes: Exploring the Neural Code*, MIT Press, Cambridge, MA.
Rinzel, J. and Ermentrout, B. (1998). Analysis of neural excitability and oscillations, *in Methods in Neuronal Modeling*, C. Koch and I. Segev (eds.), 2nd edition, The MIT Press, Cambridge, pp. 251–291.
Risken, H. (1989). *The Fokker-Planck Equation*, 2nd edition, Springer-Verlag, New York.
Roberts, W. (1994). Localization of calcium signals by a mobile calcium buffer in frog saccular hair cells, *J. Neurosci.* **14**, 3246.
Rubinow, S. (1973). *Mathematical Problems in the Biological Sciences*, S.I.A.M., Philadelphia.
Rubinow, S. (1980). Biochemical reaction theory, *in Mathematical models in molecular and cellular biology*, L. A. Segel (ed.), Cambridge Univ. Press, Cambridge, UK, pp. 13–111.
Ruoff, P. and Rensing, L. (1996). The temperature-compensated goodwin model simulates many circadian clock properties, *J. Theor. Biol.* **179**, 275–285.
Sanchez-Andres, J., Gomis, A. and Valdeolmillos, M. (1995). The electrical activity of mouse pancreatic beta-cells recorded in vivo show glucose-dependent oscillations, *J. Physiol. (London)* **486**, 223–228.
Scharf, B., Fahrner, K., Turner, L. and Berg, H. (1998). Control of direction of flagellar rotation in bacterial chemotaxis, *Proc. Nat. Acad. Sci. USA* **95**, 201–206.
Schnitzer, M. and Block, S. (1995). Statistical kinetics of processive enzymes, *Cold Spring Harbor Symposia on Quantitative Biology* **Vol. LX**, 793–802.
Segel, L. (1984). *Modeling dynamic phenomena in molecular and cellular biology*, 1st edition, Cambridge University Press, New York.
Segel, L., Chet, I. and Henis, Y. (1977). A simple quantitative assay for bacterial motility, *J. Gen. Microbiol.* **98**, 329–337.
Segev, I., Rinzel, J. and Shepard, G. (1995). *The Theoretical Foundation of Dendritic Function: Selected Papers of Wilfrid Rall with Commentaries*, MIT Press, Cambridge, MA.
Shadlen, M. and Newsome, W. (1998). The variable discharge of cortical neurons: implications for connectivity, computation, and information coding, *J. Neurosci.* **18**(10), 3870–3896.
Sherman, A. (1996). Contributions of modeling to understanding stimulus-secretion coupling in pancreatic β-cells, *Am. J. Physiol.* **271**, E362–E372.
Sherman, A. (1997). Calcium and membrane potential oscillations in pancreatic β-cells, *in Case Studies in Mathematical Modeling—Ecology, Physiology, and Cell Biology*, H. G. Othmer, F. R. Adler, M. A. Lewis and J. C. Dallon (eds.), Prentice-Hall, New Jersey, pp. 199–217.
Sherman, A., Rinzel, J. and Keizer, J. (1988). Emergence of organized bursting in clusters of pancreatic Beta cells by channel sharing., *Biophys. J.* **54**, 411–425.
Siffre, M. (1975). Six months alone in a cave, *National Geographic* **147**(3), 426–435.
Sigworth, F. (1980). The variance of sodium current fluctuations at the node of Ranvier, *J. Physiol (Lond)* **307**, 97–129.
Simon, S., Peskin, C. and Oster, G. (1992). What drives the translocation of proteins?, *Proc. Natl. Acad. Sci. USA* **89**(9), 3770–3774.
Smith, D. and Geeves, M. (1995). Strain-dependent cross-bridge cycle for muscle, *Biophys. J.* **69**(2), 524–537.
Smith, G. (1996). Analytical steady-state solution to the rapid buffering approximation near an open Ca^{2+} channel, *Biophys. J.* **71**, 3064.
Smith, G. (2001). Modeling local and global Ca^{2+} signals using reaction-diffusion equations, *in Computational Neuroscience: Realistic Modeling for Experimentalists.*, E. D. Schutter (ed.), CRC Press.
Smith, G., Dai, L., Miura, R. and Sherman, A. (2001). Asymptotic analysis buffered Ca^{2+} diffusion near a point source, *SIAM J. Appl. Math.* **61**(5), 1816–1838.

Smith, G., Keizer, J., Stern, M., Lederer, W. and Cheng, H. (1998). A simple numerical model of Ca^{2+} spark formation and detection in cardiac myocytes, *Biophys. J.* **75**, 15.

Smith, G., Wagner, J. and Keizer, J. (1996). Validity of the rapid buffering approximation near a point source for Ca^{2+} ions, *Biophys. J.* **70**, 2527.

Sneyd, J., Dale, P. and Duffy, A. (1998). Traveling waves in buffered systems: applications to calcium waves, *SIAM J. Applied Math.* **58**, 1178.

Stein, W. and Lauger, P. (1990). Kinetic properties of F0F1-ATPases, *Biophys. J.* **57**, 255–267.

Stern, B. and Nurse, P. (1996). A quantitative model for the cdc2 control of S phase and mitosis in fission yeast, *Trends Genet.* **12**, 345–350.

Stern, M. (1992). Buffering of Ca^{2+} in the vicinity of a channel pore, *Cell Calcium* **13**, 183.

Strogatz, S. (1994). *Nonlinear Dynamics and Chaos*, 1st edition, Perseus Books, Cambridge.

Surana, U., Robitsch, H., Price, C., Schuster, T., Fitch, I., Futcher, A. and Naysmyth, K. (1991). The role of CDC28 and cyclins during mitosis in the budding yeast *S. cerevisiae*, *Cell* **65**, 145–161.

Tabak, J., Senn, W., O'Donovan, M. and Rinzel, J. (2000). Modeling of spontaneous activity in the developing spinal cord using activity dependent depression in an excitatory network, *J. Neurosci.* **20**(8), 3041–3056.

Tang, Y. and Othmer, H. (1994). A model of calcium dynamics in cardiac myocytes based on the kinetics of ryanodine-sensitive calcium channels, *Biophys. J.* **67**, 2223–2235.

Tang, Y., Stephenson, J. and Othmer, H. (1996). Simplification and analysis of models of calcium dynamics based on IP_3-sensitive calcium channel kinetics, *Biophys. J.* **70**, 246–263.

Taylor, S. (1999). Deconstructing type 2 diabetes, *Cell* **97**, 9–12.

Thomas, A., Bird, G., Hajnoczky, G., Robb-Gaspers, L. and Putney Jr., J. (1996). Spatial and temporal aspects of cellular calcium signaling, *FASEB J.* **13**, 1505–1517.

Topp, B., Promislow, K., de Vries, G., Miura, R. and Finegood, D. (2000). A model of β-cell mass, insulin and glucose kinetics: Pathways to diabetes, *J. Theor. Biol.* **206**, 605–619.

Traub, R., Jefferys, J. and Whittington, M. (1999). *Fast Oscillations in Cortical Circuits*, MIT Press, Cambridge, MA.

Traub, R., Whittington, M., Stanford, I. and Jefferys, J. (1996). A mechanism for generation of long-range synchronous fast oscillations in the cortex, *Nature* **383**(6601), 621–624.

Troyer, T., Krukowski, A., Priebe, N. and Miller, K. (1998). Contrast-invariant orientation tuning in cat visual cortex: thalamocortical input tuning and correlation-based intracortical connectivity, *J. Neurosci.* **18**(15), 5908–5927.

Trussell, L. (1999). Synaptic mechanisms for coding timing in auditory neurons, *Annu. Rev. Physiol.* **61**, 477–496.

Tse, A., Tse, F. and Hille, B. (1993). Rhythmic exocytosis stimulated by GnRh-induced calcium oscillations in rat gonadotropes, *Science* **260**, 82–84.

Tuckwell, H. (1995). *Elementary Applications of Probability Theory*, 2nd edition, Chapman and Hall, New York.

Tyers, M., Tokiwa, G. and Futcher, B. (1993). Comparison of the *Saccharomyces cerevisiae* G1 cyclins: Cln3 may be an upstream activator of Cln1, Cln2 and other cyclins, *EMBO J.* **12**, 1955–1968.

Tyson, J., Hong, C., Thron, C. and Novak, B. (1999). A simple model of circadian rhythms based on dimerization and proteolysis of PER and TIM, *Biophys. J.* **77**, 2411–2417.

Tyson, J., Novak, B., Odell, G., Chen, K. and Thron, C. (1996). Chemical kinetic theory: understanding cell-cycle regulation, *Trends Biochem. Sci.* **21**, 89–96.

Valdeolmillos, M., Santos, R., Contreras, D., Soria, B. and Rosario, L. (1989). Glucose-induced oscillations of intracellular Ca^{2+} concentration resembling bursting electrical activity in single mouse islets of Langerhans, *FEBS Lett.* **259**, 19–23.

Vale, R. (2000). Millennial musings on molecular motors, *Trends Bioc. Sci.* **24**(12), M38–M42.

Van Vreeswijk, C., Abbott, L. and Ermentrout, G. (1994). When inhibition not excitation synchronizes neural firing, *J. Comput. Neurosci.* **1**(4), 313–321.
Wagner, J. and Keizer, J. (1994). Effects of rapid buffers on Ca^{2+} oscillations, *Biophys. J.* **67**, 447–456.
Wagner, J., Li, Y.-X., Pearson, J. and Keizer, J. (1998). Simulation of the fertilization calcium wave in *Xenopus laevis* eggs, *Biophys. J.* **75**, 2088–2097.
Wang, H. and Oster, G. (1998). Energy transduction in the F1 motor of ATP synthase, *Nature* **396**, 279–282.
Wang, H., Elston, T., Mogilner, A. and Oster, G. (1998). Force generation in RNA polymerase, *Biophys. J.* **74**(3), 1186–1202.
Wang, X. and Buzsaki, G. (1996). Gamma oscillation by synaptic inhibition in a hippocampal interneuronal network model, *J. Neurosci.* **16**(20), 6402–6413.
Wang, X. and Rinzel, J. (1992). Alternating and synchronous rhythms in reciprocally inhibitory model neurons, *Neural Computation* **4**, 84–97.
Warshel, A. (1991). *Computer Modeling of Chemical Reactions in Enzymes and Solutions*, Wiley, New York.
Weiss, G. (1967). First passage time problems in chemical physics, *Adv. Chem. Phys.* **13**, 1–18.
Weiss, T. (1996). *Cellular Biophysics*, 1st edition, MIT Press, Cambridge.
Welch, G. and Kell, D. (1986). Not just catalysts-molecular machines in bioenergetics, *in The Fluctuating Enzyme*, G. R. Welch (ed.), Wiley Interscience, New York, pp. 252–269.
White, J., Budde, T. and Kay, A. (1995). A bifurcation analysis of neuronal subthreshold oscillations, *Biophys. J.* **69**(4), 1203–17.
White, J., Rubinstein, J. and Kay, A. (2000). Channel noise in neurons, *TINS* **23**(3), 131–137.
Whitesell, R., Powers, A., Regen, D. and Abumrad, N. (1991). Transport and metabolism of glucose in an insulin-secreting cell line β TC-1, *Biochem.* **30**, 11560–11566.
Wilson, H. and Cowan, J. (1972). Excitatory and inhibitory interactions in localized populations of model neurons, *Biophys. J.* **12**(1), 1–24.
Winfree, A. (1987). *When Time Breaks Down*, 1st edition, Princeton University Press, Princeton.
Wong, T.-F. (1975). *Kinetics of Enzyme Mechanisms*, Academic Press, New York.
Yamada, W., Koch, C. and Adams, P. (1998). Multiple channels and calcium dynamics, *in Methods in Neuronal Modeling*, C. Koch and I. Segev (eds.), 2nd edition, The MIT Press, Cambridge, Massachusetts, pp. 251–291.
Yao, Y., Choi, J. and Parker, I. (1995). Quantal puffs of intracellular Ca^{2+} evoked by inositol trisphosphate in *Xenopus* oocytes, *J. Physiol. (Lond.)* **482**, 533.
Yasuda, R., Noji, H., Kinosita, K. and Yoshida, M. (1998). F1-ATPase is a highly efficient molecular motor that rotates with discrete 120° steps, *Cell* **93**, 1117–1124.
Zigmond, M., Bloom, F., Landis, S., Roberts, J. and Squire, L. (1999). *Fundamental Neuroscience*, Academic Press, New York.

Index