SOLUTION OF SECOND ORDER DIFFERENTIAL EQUATIONS USING THE GODUNOV INTEGRATION METHOD

by

Christopher Paul Beamis

A Thesis Submitted to the Faculty of the
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
In Partial Fulfillment of the Requirements
For the Degree of
MASTER OF SCIENCE
WITH A MAJOR IN ELECTRICAL ENGINEERING
In the Graduate College
THE UNIVERSITY OF ARIZONA

1 9 9 0

STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgement the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: Christopher Paul Beam

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

F. François E. Cellier
F. E. Cellier
Associate Professor of
Electrical and Computer Engineering

July 18, 1990
Date

ACKNOWLEDGMENTS

I would like to thank Dr. François Cellier, my major advisor, for guidance, assistance, general support and for being extremely patient with me.

I would also like to thank Dr. Bruce Bayly for helping me through the math details which I would have otherwise become bogged down in.

I would like to thank Dr. Olgierd Palusinski and Dr. Bruce Bayly for taking the time to participate on my thesis committee.

Finally, I would like to express my gratitude to Mike Roehm of Hughes Aircraft Company who helped me get the fellowship which allowed me to attend graduate school in the first place.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

ABSTRACT

This MS Thesis proposes a method for the direct numerical solution of systems of second order differential equations. Most numerical integration techniques currently in use rely on separating each second order equation into two first order equations. This thesis investigates an alternative integration technique proposed by Godunov which solves second order differential equations directly. It is hoped that the Godunov method may provide an advantage over first order system solvers in the areas of numerical stability domains, truncation errors and the number of floating point operations required for a solution.

A numerical stability domain will be developed for the Godunov method both analytically and numerically, and expressions will be developed for the truncation error which results from using the Godunov method to solve the wave equation, a mechanical model of the human body and a seventh order passive electrical circuit. It will be shown that the Godunov method compares favorably against the Adams-Bashforth third order method when used to solve both the mechanical system and the hyperbolic partial differential equation, but that there are potential problems when this method is used to simulate electrical circuits which result in integro-differential equations.

CHAPTER 1 - INTRODUCTION

It is not an unusual occurance for continuous systems to be expressed naturally in terms of second order ordinary differential equations. This happens with problems in mechanics, hydrodynamics, electrical circuits and others. For example, every time a mechanical system has forces being applied to masses, a second order differential equation will result due to the application of Newton's law $F = ma = m\ddot{x}$. A simple example of this phenomenon is shown in the mass, linear spring and linear damper illustrated in Figure (1.1).

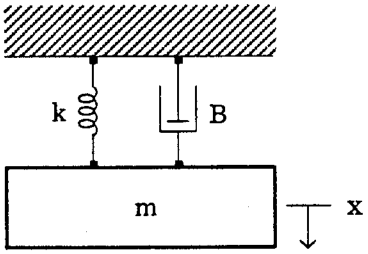


Figure 1.1 Simple Translational Mechanical System

Using the relationships

$$F_{mass} = ma = m\frac{d^2x(t)}{dt^2}$$

$$F_{damper} = Bv = B\frac{dx(t)}{dt}$$

$$F_{spring} = kx(t) \quad ,$$

the equation

$$\frac{d^2x(t)}{dt^2} = \frac{F}{m} = \frac{mg - kx(t) - B\frac{dx(t)}{dt}}{m}$$

results, which has the form $\ddot{x}(t) = f(x(t), \dot{x}(t), t)$. Another simple example of a mechanical system which will result in a second order differential equation is shown in Figure (1.2).

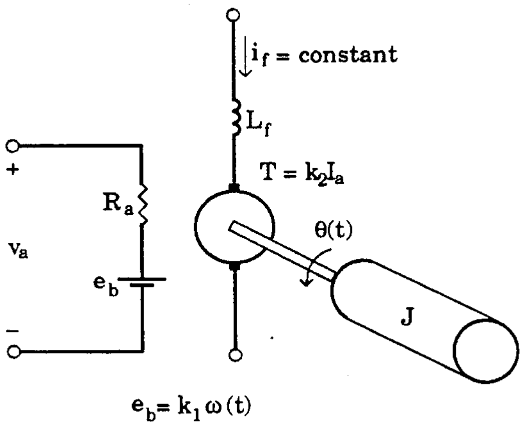


Figure 1.2 Simple Rotational Mechanical System

If the angle θ is the variable of interest in this case then a second order differential equation will result due to the relationship between motor torque and load inertia,

$$T_m(t) = J \frac{d^2\theta(t)}{dt^2} \quad .$$

This system will result in the differential equation

$$\frac{d^2\theta(t)}{dt^2} = \frac{-k_2 k_1 \frac{d\theta(t)}{dt} + k_2 v_a(t)}{J R_a}$$

which fits the form of

$$\ddot{x}(t) = f(\dot{x}(t), u(t), t) \quad .$$

Any electrical system which has nodes or loops which involve both capacitors and inductors will result in integro-differential equations which can be differentiated to give second order differential equations. A simple example is shown in Figure (1.3).

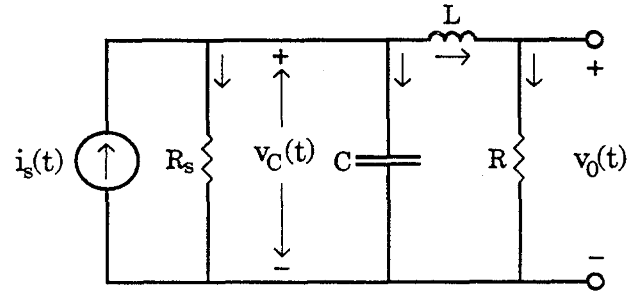


Figure 1.3 Simple Passive Electrical System

Applying Kirchoff's Current Law to both nodes of this circuit results in the equations

$$i_s(t) - C\frac{dv_C(t)}{dt} - \frac{1}{L}\int (v_C(t) - v_o(t))dt - \frac{1}{R_s}v_C(t) = 0 \tag{1.1}$$

$$\frac{1}{L}\int (v_C(t) - v_o(t))dt - \frac{1}{R}v_o(t) = 0 \quad . \tag{1.2}$$

Differentiating equations (1.1) and (1.2) and solving each for the highest order derivative results in a combination of 1st and 2nd order differential equations

$$\begin{aligned} \frac{d^2v_C(t)}{dt^2} &= -\frac{1}{LC}v_C(t) + \frac{1}{LC}v_o(t) - \frac{1}{R_sC}\frac{dv_C(t)}{dt} + \frac{1}{C}\frac{di_s(t)}{dt} \\ \frac{dv_o(t)}{dt} &= \frac{R}{L}v_C(t) - \frac{R}{L}v_o(t) \quad . \end{aligned} \tag{1.3}$$

Equations (1.3) fit the general form

$$\ddot{x}_1(t) = f_1(x_1(t), \dot{x}_1(t), x_2(t), \dot{u}(t), t)$$

$$\dot{x}_2(t) = f_2(x_1(t), x_2(t), t) \quad .$$

Second order partial differential equations, which often arise from problems in hydrodynamics, can be solved by the use of difference equations to discretize one of the variables and then solving the partial derivative of the other variable by using numerical techniques for ordinary differential equations. An example of this is the wave equation

$$\frac{\partial^2 U}{\partial t^2} = \frac{\partial^2 U}{\partial x^2} \quad . \tag{1.4}$$

If the x-axis is discretized using the first order centered difference scheme

$$\frac{\partial^2 U}{\partial x^2} = \frac{1}{(\Delta x)^2}[U(x+\Delta x) - 2U(x) + U(x-\Delta x)] \quad ,$$

then the left hand side of equation (1.4) can be conveniently formulated as a 2nd order ordinary differential equation, giving

$$\frac{d^2 U}{dt^2} = \frac{1}{(\Delta x)^2}[U(x+\Delta x) - 2U(x) + U(x-\Delta x)] \quad .$$

If the x-axis is separated into N discrete intervals, then a system of N 2nd order ordinary differential equations will result from this partial differential equation.

Most ordinary differential equation numerical techniques currently in use rely on separating each 2nd order differential equation

$$\frac{d^2 x(t)}{dt^2}$$

into two first order differential equations so that

$$\begin{aligned} \frac{dx_1(t)}{dt} &= x_2(t) \\ \text{and} \quad \frac{dx_2(t)}{dt} &= \frac{d^2 x(t)}{dt^2} \quad , \end{aligned}$$

and then solving the resulting system of 1st order differential equations. Godunov, however, came up with a method that allows the 2nd order differential equation to be solved directly without having to separate it into twice as many 1st order differential equations [3]. This method is essentially a time domain version of the centered difference scheme used to discretize the x-axis of equation (1.4).

$$x_{n+1} = 2x_n - x_{n-1} + h^2 \ddot{x}_n \quad .$$

Since this integration technique involves the evaluation of only half as many differential equations as does a first order differential equation technique, it should take fewer floating point operations per step. In Chapter 3, experimental results, the Godunov method is compared to the Adams-Bashforth 3rd order method for several examples. The Godunov method should have an additional advantage in the number of floating point operations required per step over the Adams-Bashforth 3rd order method due to the fact that each Adams-Bashforth 3rd order step involves the summation of 4 vectors plus 4 multiplications of scalars and vectors, while each Godunov step involves the summation of only 3 vectors plus 2 multiplications of vectors and scalars. In addition, the Godunov method will be working on vectors with half as many elements as those that the Adams-Bashforth method will have.

In Chapter 2 the Godunov method is derived showing the local truncation error when used to solve

$$\ddot{\underline{x}}(t) = \underline{f}(\underline{x}(t), t) \quad .$$

Then, a stability domain is found analytically for the Godunov method when used to solve this same equation, and a stability domain is found numerically when the Godunov method is used to solve the equation

$$\ddot{\underline{x}}(t) = \underline{f}(\underline{x}(t), \dot{\underline{x}}(t), t) \quad .$$

Stability domains and truncation errors are then found for the four systems

$$1)\quad \underline{\ddot{x}}(t) = -\omega^2 \underline{x}(t)$$

$$2)\quad \underline{\ddot{x}}(t) = -\omega^2 \underline{x}(t) - C\underline{\dot{x}}(t) + \underline{d}u(t)$$

$$3)\quad \underline{\ddot{x}}_1(t) = A\underline{x}_1(t) + B\underline{\dot{x}}_1(t) + \underline{c}x_2(t) + \underline{d}\dot{u}(t)$$

$$x_2(t) = \underline{e}'\underline{x}_1(t) + fx_2(t)$$

$$4)\quad \underline{\ddot{x}}(t) = \underline{f}(\underline{x}(t), t) \quad .$$

These four systems are then solved analytically and numerically in Chapter 3 and results are compared with the predictions of Chapter 2. Chapter 4 discusses the advantages and disadvantages of the Godunov technique when compared with 1st order differential equation solvers. The experimental results of Chapter 3 are used in this chapter to give the reader an idea of when the Godunov integration technique might be the method of choice and when 1st order differential equation solvers might be the way to go.

CHAPTER 2 - ANALYTIC RESULTS

There are four specific systems which will be examined analytically in this chapter. Stability and truncation error will be analyzed for three types of systems:

1) $\ddot{\underline{x}}(t) = \underline{f}(\underline{x}(t),t)$ (wave equation and sine-Gordon equation) (2.1)

2) $\ddot{\underline{x}}(t) = \underline{f}(\underline{x}(t),\dot{\underline{x}}(t),\underline{u}(t),t)$ (passive mechanical system) (2.2)

3) $\ddot{\underline{x}}_1(t) = \underline{f}_1(\underline{x}_1(t),\dot{\underline{x}}_1(t),\underline{x}_2(t),\underline{u}(t),t)$ (passive electrical system) (2.3)

$\dot{\underline{x}}_2(t) = \underline{f}_2(\underline{x}_1(t),\dot{\underline{x}}_1(t),\underline{x}_2(t),\underline{u}(t),t)$.

First, a derivation of Godunov's method for solving the second order problem $\ddot{\underline{x}}(t) = \underline{f}(\underline{x}(t),t)$ will be shown. If we expand $x(t)$ in a Taylor series about $x(t-\Delta t)$, we obtain

$$\underline{x}(t) = \underline{x}(t-\Delta t) + \Delta t\dot{\underline{x}}(t-\Delta t) + \frac{(\Delta t)^2}{2}\ddot{\underline{x}}(t-\Delta t) + \frac{(\Delta t)^3}{3!}\underline{x}^{(iii)}(t-\Delta t) + O((\Delta t)^4) \quad . \tag{2.4}$$

Similarly, expanding $x(t-2\Delta t)$ in a Taylor series about $x(t-\Delta t)$ results in the expression

$$\begin{aligned}\underline{x}(t-2\Delta t) =& \underline{x}(t-\Delta t) - \Delta t\dot{\underline{x}}(t-\Delta t) + \frac{(\Delta t)^2}{2}\ddot{\underline{x}}(t-\Delta t) \\ &- \frac{(\Delta t)^3}{3!}\underline{x}^{(iii)}(t-\Delta t) + O((\Delta t)^4) \quad .\end{aligned} \tag{2.5}$$

Adding equations (2.4) and (2.5) gives the formula:

$$\underline{x}(t) + \underline{x}(t-2\Delta t) = 2\underline{x}(t-\Delta t) + (\Delta t)^2\ddot{\underline{x}}(t-\Delta t) + O((\Delta t)^4)$$

$$\text{or} \quad \underline{x}(t) = 2\underline{x}(t-\Delta t) - \underline{x}(t-2\Delta t) + (\Delta t)^2\ddot{\underline{x}}(t-\Delta t) + O((\Delta t)^4) \tag{2.6}$$

which is a centered difference scheme in time and has no first derivative terms. Given a system of the form of equation (2.1), the second derivative $\ddot{\underline{x}}(t) = f(\underline{x}(t), t)$ can now be explicitly substituted into equation (2.6), giving the formula

$$\underline{x}(t) = 2\underline{x}(t-\Delta t) - \underline{x}(t-2\Delta t) + (\Delta t)^2 f(\underline{x}(t-\Delta t), t-\Delta t) + O((\Delta t)^4)$$

which gives third order global accuracy. If equation (2.1) were to be separated into twice as many first order equations, each step of a first order equation solver would have vectors twice the size to deal with, thus resulting in approximately twice the number of floating point operations required per step if the number of arithmetic operations required in the first order solver itself were approximately the same as those in the second order solver. Because of this, it is hoped that the Godunov method will require far fewer floating point operations than will a first order differential equation solver.

For equations (2.2) and (2.3), a hybrid combination of first and second order solvers will have to be used. Equation (2.2) will require a first order equation solver to extrapolate $\dot{\underline{x}}(t)$ forward in time because $\dot{\underline{x}}(t)$ is required explicitly in the right hand side of the differential equation. If the Adams-Bashforth third order method is used for the first derivative $\dot{\underline{x}}(t)$, and the centered difference scheme (Godunov's method) is used for $\underline{x}(t)$, two equations will be solved simultaneously:

$$\begin{aligned}
\underline{x}(t+\Delta t) =& 2\underline{x}(t) - \underline{x}(t-\Delta t) + (\Delta t)^2 \underline{f}(\underline{x}(t), \dot{\underline{x}}(t), \underline{u}(t), t)\\
\dot{\underline{x}}(t+\Delta t) =& \dot{\underline{x}}(t) + \frac{(\Delta t)}{12}[23\underline{f}(\underline{x}(t), \dot{\underline{x}}(t), \underline{u}(t), t)\\
&-16\underline{f}(\underline{x}(t-\Delta t), \dot{\underline{x}}(t-\Delta t), \underline{u}(t-\Delta t), t-\Delta t)\\
&+5\underline{f}(\underline{x}(t-2\Delta t), \dot{\underline{x}}(t-2\Delta t), \underline{u}(t-2\Delta t), t-2\Delta t)] \quad .
\end{aligned}$$

Systems of this type will not offer quite the same advantages in terms of floating point operations counts by the use of the Godunov method over a first order equation solver because solutions for both the state vector $\underline{x}(t)$ and its first derivative $\dot{\underline{x}}(t)$ are required.

A hybrid method will also be needed for systems of the type of equation (2.3). A Godunov step will be used to extrapolate $\underline{x}_1(t)$, and a first order system solver can be used to solve for $\dot{\underline{x}}_1(t)$, and $\underline{x}_2(t)$. This will result in the simultaneous solution of three equations:

$$
\begin{aligned}
\underline{x}_1(t+\Delta t) =& 2\underline{x}_1(t) - \underline{x}_1(t-\Delta t) + (\Delta t)^2 \underline{f}_1(\underline{x}_1(t), \dot{\underline{x}}_1(t), \underline{x}_2(t), \underline{u}(t), t) \\
\dot{\underline{x}}_1(t+\Delta t) =& \dot{\underline{x}}_1(t) + \frac{(\Delta t)}{12}[23\underline{f}_1(\underline{x}_1(t), \dot{\underline{x}}_1(t), \underline{x}_2(t), \underline{u}(t), t) \\
& -16\underline{f}_1(\underline{x}_1(t-\Delta t), \dot{\underline{x}}_1(t-\Delta t), \underline{x}_2(t-\Delta t), \underline{u}(t-\Delta t), t-\Delta t) \\
& +5\underline{f}_1(\underline{x}_1(t-2\Delta t), \dot{\underline{x}}_1(t-2\Delta t), \underline{x}_2(t-2\Delta t), \underline{u}(t-2\Delta t), t-2\Delta t)] \\
\underline{x}_2(t+\Delta t) =& \underline{x}_2(t) + \frac{(\Delta t)}{12}[23\underline{f}_2(\underline{x}_1(t), \dot{\underline{x}}_1(t), \underline{x}_2(t), \underline{u}(t), t) \\
& -16\underline{f}_2(\underline{x}_1(t-\Delta t), \dot{\underline{x}}_1(t-\Delta t), \underline{x}_2(t-\Delta t), \underline{u}(t-\Delta t), t-\Delta t) \\
& +5\underline{f}_2(\underline{x}_1(t-2\Delta t), \dot{\underline{x}}_1(t-2\Delta t), \underline{x}_2(t-2\Delta t), \underline{u}(t-2\Delta t), t-2\Delta t)] \quad .
\end{aligned}
$$

Since this sytem type will require the solution of $\underline{x}_1(t)$, $\dot{\underline{x}}_1(t)$ and $\underline{x}_2(t)$, the floating point operations count advantage over first order systems solvers will decrease even further.

The remainder of chapter two will be devoted first to an analytical stability analysis and then a numerical stability analysis of the Godunov method and then to the development of the differential equations arising from each specific system to be studied and to a stability analysis and a truncation error analysis for each of these four systems.

ANALYTIC STABILITY ANALYSIS FOR THE GODUNOV METHOD

A range of values of the step size h for which the Godunov step, when used to solve the equation

$$\ddot{\underline{x}}(t) = \underline{f}(\underline{x}(t), t) \tag{2.7}$$

is stable can be found by using the following argument [7]. Denoting the analytic solution at time t_n by $\underline{x}(t_n)$ and the Godunov solution at the n^{th} step by $\hat{\underline{x}}_n$, we can write

$$\underline{x}(t_{n+2}) - 2\underline{x}(t_{n+1}) + \underline{x}(t_n) = h^2 \underline{f}(\underline{x}(t_{n+1}), t_{n+1}) + \underline{T}_{n+2} \tag{2.8}$$

$$\hat{\underline{x}}_{n+2} - 2\hat{\underline{x}}_{n+1} + \hat{\underline{x}}_n = h^2 \underline{f}(\hat{\underline{x}}_{n+1}, t_{n+1}) + \underline{R}_{n+2} \tag{2.9}$$

where $\underline{T}_n$ and $\underline{R}_n$ are the truncation error and the roundoff error respectively at the n^{th} step. Subtracting (2.9) from (2.8) to get an expression for the error results in

$$\begin{aligned}&(\underline{x}(t_{n+2}) - \hat{\underline{x}}_{n+2}) - (2\underline{x}(t_{n+1}) - 2\hat{\underline{x}}_{n+1}) + (\underline{x}(t_n) - \hat{\underline{x}}_n)\\ &= h^2(\underline{f}(\underline{x}(t_{n+1}), t_{n+1}) - \underline{f}(\hat{\underline{x}}_{n+1}, t_{n+1})) + \underline{\phi}_{n+2}\end{aligned} \tag{2.10}$$

where $\underline{\phi} = \underline{T} - \underline{R}$. The mean value theorem for derivatives states that

$$\frac{\underline{f}(\underline{x}(t_{n+1}), t_{n+1}) - \underline{f}(\hat{\underline{x}}_{n+1}, t_{n+1})}{\underline{x}(t_{n+1}) - \hat{\underline{x}}_{n+1}} = \frac{\partial \underline{f}(\underline{\varsigma}_{n+1}, t_{n+1})}{\partial \underline{x}} \tag{2.11}$$

for some vector $\underline{\varsigma}_{n+1}$ lying between $\underline{x}(t_{n+1})$ and $\hat{\underline{x}}_{n+1}$. If we solve equation (2.11) for $f(\underline{x}(t_{n+1}), t_{n+1}) - f(\hat{\underline{x}}_{n+1}, t_{n+1})$, and then substitute the result into equation (2.10), we get

$$\hat{\underline{e}}_{n+2} - 2\hat{\underline{e}}_{n+1} + \hat{\underline{e}}_n = h^2 \frac{\partial f(\underline{\varsigma}_{n+1}, t_{n+1})}{\partial x} \hat{\underline{e}}_{n+1} + \underline{\phi}_{n+2} \tag{2.12}$$

where $\hat{\underline{e}}_n = \underline{x}(t_n) - \hat{\underline{x}}_n$. If equation (2.7) is linear and time invariant, that is,

$$\ddot{\underline{x}}(t) = \underline{f}(\underline{x}(t), t) = -\omega^2 \underline{x}(t)$$

then

$$\frac{\partial \underline{f}}{\partial \underline{x}} = -\omega^2 \quad .$$

Using this fact, and making one assumption, namely $\underline{\phi}_n = \underline{\phi}$, constant, equation (2.12) becomes

$$\underline{\hat{e}}_{n+2} - 2\underline{\hat{e}}_{n+1} + \omega^2 h^2 \underline{\hat{e}}_{n+1} + \underline{\hat{e}}_n - \underline{\phi} = 0 \quad . \tag{2.13}$$

If we let $-\omega^2 \underline{\xi}_j = -\lambda_j^2 \underline{\xi}_j$ where $\underline{\xi}_j$ and λ_j are the j^{th} eigenvector and eigenvalue respectively of ω, and let $\underline{\hat{e}}_n = C_n \underline{\xi}_j$, then equation (2.13) becomes

$$C_{n+2}\underline{\xi}_j - 2C_{n+1}\underline{\xi}_j + \omega^2 h^2 C_{n+1}\underline{\xi}_j + C_n\underline{\xi}_j - \underline{\phi} = 0$$

$$\text{or} \quad C_{n+2}\underline{\xi}_j - 2C_{n+1}\underline{\xi}_j + \lambda_j^2 \underline{\xi}_j h^2 C_{n+1} + C_n\underline{\xi}_j - \underline{\phi} = 0 \quad .$$

Letting $\underline{\phi}$ go to zero (a reasonable assumption for small h due to the fact that the Godunov local truncation error is $O(h^4)$ and machine roundoff error is on the order of 10^{-17}), the $\underline{\xi}_j{}^{\underline{s}}$ drop out and we get

$$C_{n+2} - (2 - \lambda_j^2 h^2)C_{n+1} + C_n = 0 \quad .$$

Assuming a solution $C_n = r^n$ results in the quadratic equation

$$r^2 - (2 - \lambda_j^2 h^2)r + 1 = 0 \quad .$$

The solution to this equation is

$$r = 1 - \frac{1}{2}\lambda_j^2 h^2 \pm \sqrt{\frac{-\lambda_j h^2 (4 - \lambda_j^2 h^2)}{4}} \quad .$$

If $4 - \lambda_j h^2 > 0$, then

$$r = 1 - \frac{1}{2}\lambda_j^2 h^2 \pm i\lambda_j h \sqrt{1 - \frac{\lambda_j h^2}{4}}$$

$$\text{and} \quad |r|^2 = [1 - \frac{1}{2}\lambda_j^2 h^2]^2 + \lambda_j^2 h^2 [1 - \frac{\lambda_j^2 h^2}{4}] \quad ,$$

$$\text{or} \quad |r|^2 = 1 - \lambda_j^2 h^2 + \frac{\lambda_j^4 h^4}{4} + \lambda_j^2 h^2 - \frac{\lambda^4 h^4}{4} = 1 \quad ,$$

$$\text{so} \quad |r| = 1 \quad \text{if} \quad 4 - \lambda_j^2 h^2 > 0, \qquad \text{or} \qquad h < \frac{2}{\lambda_j} \quad .$$

It can be shown that this will result in real, positive h because λ_j will always be real and positive for the bounded, linear, time invariant case of equation (2.7). Consider the eigenvectors and eigenvalues of the matrix ω which will solve $-\omega^2\underline{\xi}_j = -\lambda_j{}^2\underline{\xi}_j$. If the solution to equation (2.7), $\underline{x}(t)$, is stable when an eigenvector, $\underline{\xi}_j$, is used as the initial conditions, then the solution will also be stable when a linear combination of eigenvectors is used for the initial conditions. Since any vector can be expressed as a linear combination of eigenvectors, a single eigenvector can be used to check the stability of equation (2.7). This can be done by replacing the right hand side of $\ddot{\underline{x}}(t) = -\omega^2\underline{x}(t)$ with $-\lambda_j{}^2\underline{x}(t)$, giving

$$\ddot{\underline{x}}(t) = -\lambda_j{}^2\underline{x}(t) \quad . \tag{2.14}$$

Now, if λ_j is real and negative, equation (2.14) has the solution

$$\underline{x}(t) = \underline{c}_1 e^{\lambda_j t} + \underline{c}_2 e^{-\lambda_j t}$$

which is unbounded as t approaches infinity. If λ_j is complex, then equation (2.14) has the solution

$$\underline{x}(t) = (\underline{c}_R + \mathrm{i}\underline{c}_I)e^{\mathrm{i}\lambda_j t} + (\underline{c}_R - \mathrm{i}\underline{c}_I)e^{-\mathrm{i}\lambda_j t} \tag{2.15}$$

which becomes

$$\underline{x}(t) = (\underline{c}_R + \mathrm{i}\underline{c}_I)e^{-\lambda_{jI}t}(cos\lambda_{jR}t + \mathrm{i}sin\lambda_{jR}t) + (\underline{c}_R - \mathrm{i}\underline{c}_I)e^{\lambda_{jI}t}(cos\lambda_{jR}t - \mathrm{i}sin\lambda_{jR}t) \tag{2.16}$$

by the use of the Euler identity $e^{\mathrm{u}+\mathrm{iv}} = e^{\mathrm{u}}(cos\mathrm{v} + \mathrm{i}sin\mathrm{v})$. From equations (2.15) and (2.16), the solution to (2.14) will clearly be unstable if the imaginary part of λ_j is not equal to zero or if λ_j is real and negative. For real and positive λ_j, the solution to equation (2.14) is

$$\begin{aligned} \underline{x}(t) &= 2[\underline{c}_R cos\lambda_{jR}t - \underline{c}_I sin\lambda_{jR}] \\ &= 2\underline{c}_R cos\lambda_{jR}t \quad \text{because } \underline{c}_I = \underline{0} \text{ if } \lambda_{jI} = 0 \ . \end{aligned}$$

So for bounded solutions to the linear, time-invariant case of (2.7), λ_j will be real and positive, and the value of h necessary for a stable solution using the Godunov method will satisfy the relationship

$$h < \frac{2}{\lambda_j} \quad .$$

A NUMERICAL ANALYSIS OF THE GODUNOV STABILITY DOMAIN

A numerical analysis can be used to determine the stability domain of the Godunov method in terms of the eigenvalues of the system to be solved (for linear systems of the type $\ddot{\underline{x}}(t) = f(\underline{x}(t), \dot{\underline{x}}(t), t)$). If all the eigenvalues of the system in question have a magnitude of one, then the limits of stability $|\lambda h|$ will be equal to the magnitude of the largest value of h allowed for stability at that particular angle in the complex plane defined by the complex eigenvalue. For example, the equation

$$\frac{d^2x(t)}{dt^2} + 2\sigma\frac{dx(t)}{dt} + (\sigma^2 + \omega^2)x(t) = 0$$

has eigenvalues at $-\sigma \pm j\omega$. σ and ω can be chosen at different locations on the unit circle resulting in a value of $|\lambda| = 1$. If this equation is solved using a Godunov step for $x(t)$ and an Adams-Bashforth 3rd order step for $\dot{x}(t)$ at each of the values of σ and ω that we choose to be on the unit circle, then the maximum value of h for which the numerical system

$$\begin{aligned}
x_{n+1} &= 2x_n - x_{n-1} + h^2[-2\sigma\dot{x}_n - (\sigma^2 + \omega^2)x_n] \\
\dot{x}_{n+1} &= \dot{x}_n + \frac{h}{12}[23(-2\sigma\dot{x}_n - (\sigma^2 + \omega^2)x_n) \\
&\quad - 16(-2\sigma\dot{x}_{n-1} - (\sigma^2 + \omega^2)x_{n-1}) \\
&\quad + 5(-2\sigma\dot{x}_{n-2} - (\sigma^2 + \omega^2)x_{n-2})]
\end{aligned}$$

is stable is equal to the limit of stability $|\lambda h|$ at that angle of $-\sigma \pm j\omega$. The stability domains found using this technique (using the CTRL-C code found in Appendix A) for both the Godunov method and the Adams-Bashforth 3rd order method are shown in Figure (2.1).

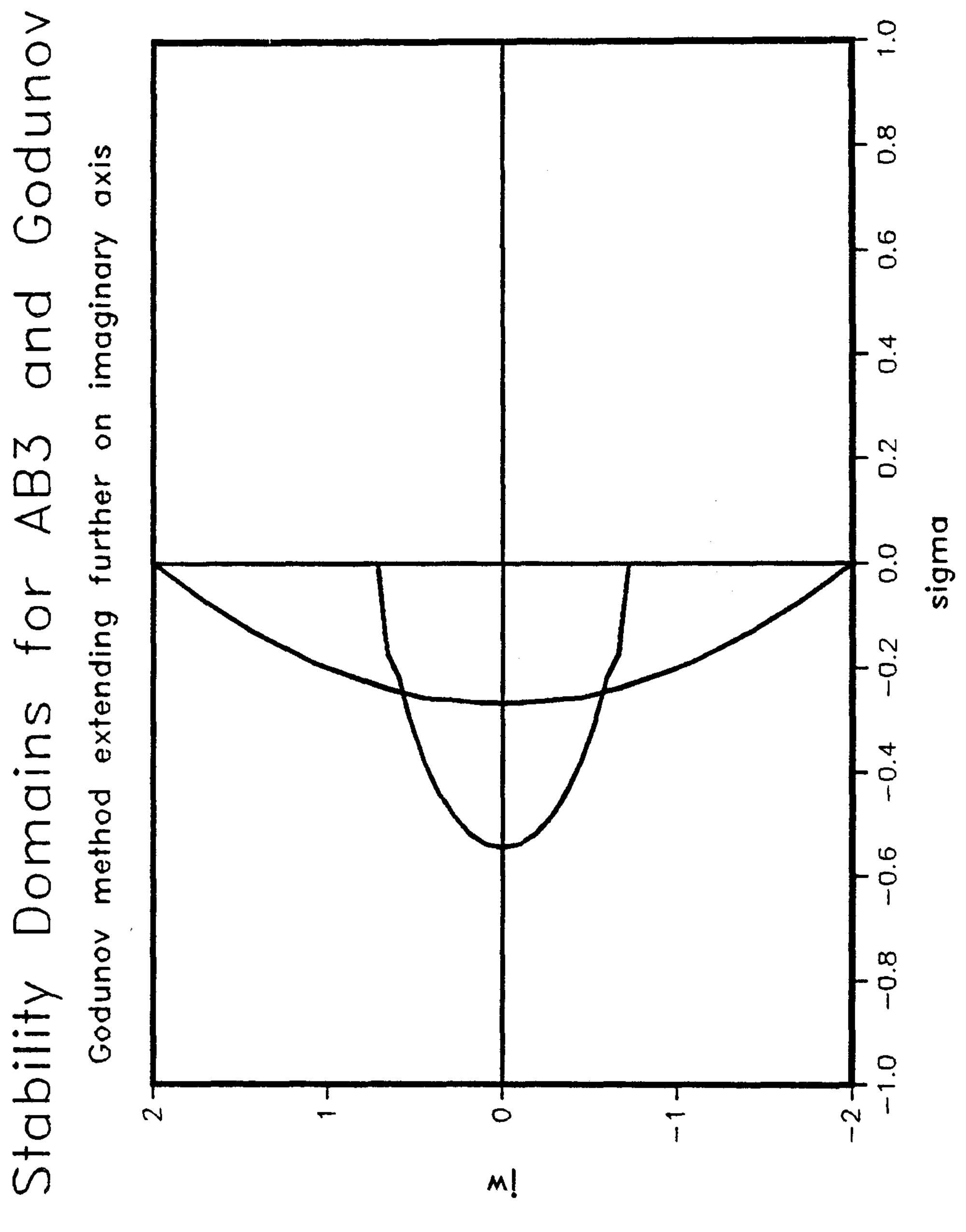


Figure 2.1 Stability Domains for AB3 and Godunov

This equation was considered stable as long as the eigenvalues of the numerical system all had absolute values less than or equal to one. Each example chosen to test the Godunov integration technique is examined more closely in the remainder of Chapter 2.

FIRST EXAMPLE: A HYPERBOLIC PARTIAL DIFFERENTIAL EQUATION (WAVE EQUATION)

$$\frac{\partial^2 U}{\partial t^2} = \frac{\partial^2 U}{\partial x^2}$$

Initial conditions:

$$\frac{\partial U(x,0)}{\partial t} = 0 \quad , \quad U(x,0) = sin(\frac{\pi}{2}), \qquad (2.17, 2.18)$$

and boundary conditions:

$$U(0,t) = 0 \quad , \quad \frac{\partial U(1,t)}{\partial x} = 0$$

are used for this example. The hyperbolic pde is solved numerically by discretizing the x-axis using a centered difference formula and then discretizing each x element in time with the Godunov step. For this example the x-axis was separated into 10 discrete sections of length $\Delta x = 0.1$ each. The equation,

$$\frac{\partial^2 U_j}{\partial x^2} = \frac{1}{(\Delta x)^2}[U_{j+1} - 2U_j + U_{j-1}] \quad ,$$

results from using a first order centered difference scheme to discretize in space. The boundary elements,

$$U(0,t) \qquad \text{and} \qquad \frac{\partial U(0,t)}{\partial t} \quad ,$$

are not calculated explicitly since they have already been defined, and a symmetry condition,

$$U(1+\Delta x, t) = U(1-\Delta x, t) \quad ,$$

is used at the x=1 boundary. If the centered difference equations are left in second derivative form, ten second order ordinary differential equations result:

$$\begin{aligned}
\frac{d^2U_1}{dt^2} &= \frac{1}{(\Delta x)^2}[U_2 - 2U_1] \\
\frac{d^2U_2}{dt^2} &= \frac{1}{(\Delta x)^2}[U_3 - 2U_2 + U_1] \\
&\vdots \\
\frac{d^2U_9}{dt^2} &= \frac{1}{(\Delta x)^2}[U_{10} - 2U_9 + U_8] \\
\frac{d^2U_{10}}{dt^2} &= \frac{1}{(\Delta x)^2}[2U_9 - 2U_{10}] \quad .
\end{aligned}$$

This set of second order equations fits the general form

$$\ddot{\underline{x}}(t) = -\omega^2 \underline{x}(t) \tag{2.19}$$

where (for this example)

$$\omega^2 = \begin{pmatrix} 200 & -100 & 0 & 0 & \dots & 0 \\ -100 & 200 & -100 & 0 & \dots & 0 \\ 0 & -100 & 200 & -100 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & -200 & 200 \end{pmatrix} \quad ,$$

and $\underline{x}(t) = \underline{U}(t)$. The space discretized equations can also be solved as a set of first order ordinary differential equations by separating each second partial derivative with respect to time into two first derivatives:

$$\begin{aligned}
\frac{\partial U_j}{\partial t} &= V_j \quad , \\
\frac{\partial V_j}{\partial t} &= \frac{\partial^2 U_j}{\partial x^2} = \frac{1}{(\Delta x)^2}[U_{j+1} - 2U_j + U_{j-1}] \quad .
\end{aligned}$$

This results in the following set of twenty 1st order ordinary differential equations:

$$
\begin{aligned}
\frac{dU_1}{dt} &= V_1 \\
\frac{dV_1}{dt} &= \frac{1}{(\Delta x)^2}[U_2 - 2U_1] \\
\frac{dU_2}{dt} &= V_2 \\
\frac{dV_2}{dt} &= \frac{1}{(\Delta x)^2}[U_3 - 2U_2 + U_1] \\
&\vdots \\
\frac{dU_9}{dt} &= V_9 \\
\frac{dV_9}{dt} &= \frac{1}{(\Delta x)^2}[U_{10} - 2U_9 + U_8] \\
\frac{dU_{10}}{dt} &= V_{10} \\
\frac{dV_{10}}{dt} &= \frac{1}{(\Delta x)^2}[2U_9 - 2U_{10}] \quad .
\end{aligned}
$$

This set of first order ordinary differential equations can then be solved numerically using a first order differential equation solver such as Adams-Bashforth.

Stability Analysis for Godunov Integration of the Wave Equation

Since the wave equation results in a linear, time-invariant system (equation 2.19), the results of the analytic stability analysis for the Godunov method can be applied directly, namely that the Godunov solution of the wave equation will be bounded for

$$h < \frac{2}{\lambda_j} \quad .$$

Truncation Error Analysis

In general, discretizing the wave equation

$$\frac{\partial^2 U}{\partial t^2} = \frac{\partial^2 U}{\partial x^2} \quad ,$$

using a Godunov step in the time axis and a first order centered difference scheme in the x-axis results in the following equation:

$$\frac{\partial^2 U}{\partial t^2} = \frac{U_{i+1,j} - 2U_{i,j} + U_{i-1,j}}{(\Delta t)^2} = \frac{U_{i,j+1} - 2U_{i,j} + U_{i,j-1}}{(\Delta x)^2} = \frac{\partial^2 U}{\partial x^2}, \qquad (2.22)$$

where $U_{i,j}$ represents U at the i^{th} time step and the j^{th} space step. Solving equation (2.22) for U at space step j and time step i+1 gives

$$\begin{aligned} U_{i+1,j} &= 2U_{i,j} - U_{i-1,j} + \frac{(\Delta t)^2}{(\Delta x)^2}[U_{i,j+1} - 2U_{i,j} + U_{i,j-1}] \\ &\quad + T_{i+1,j} + R_{i+1,j} \\ \text{or} \qquad U(x,t+\Delta t) &= 2U(x,t) - U(x,t-\Delta t) \\ &\quad + \frac{(\Delta t)^2}{(\Delta x)^2}[U(x+\Delta x,t) - 2U(x,t) + U(x-\Delta x,t)] \\ &\quad + T(x,t+\Delta t) + R(x,t+\Delta t) \quad . \end{aligned}$$

where T is the truncation error and R is the roundoff error. An expression for the truncation error is then given by

$$\begin{aligned} T(x,t+\Delta t) =& U(x,t+\Delta t) - 2U(x,t) + U(x,t-\Delta t) \\ & - \frac{(\Delta t)^2}{(\Delta x)^2}[U(x+\Delta x,t) - 2U(x,t) + U(x-\Delta x,t)] \\ & - R(x,t+\Delta t) \quad . \end{aligned} \qquad (2.23)$$

The general solution to the wave equation is given by:

$$U(x,t) = f(x-t) + g(x+t) \qquad (2.24)$$

or $U(x,t+\Delta t) = f(x-(t+\Delta t)) + g(x+(t+\Delta t))$ at time $t+\Delta t$. Substituting the analytic solution (2.24) into the right hand side of equation (2.23) yields an expression for the local truncation error explicitly in terms of the general solution:

$$\begin{aligned}
&|T(x,t+\Delta t)| = \\
&|f(x-(t+\Delta t)) + g(x+(t+\Delta t)) - 2f(x-t) - 2g(x+t) \\
&+f(x-(t-\Delta t)) + g(x+(t-\Delta t)) \\
&-\frac{(\Delta t)^2}{(\Delta x)^2}[f(x+\Delta x-t) + g(x+\Delta x+t) \\
&-2f(x-t) - 2g(x+t) + f(x-\Delta x-t) + g(x-\Delta x+t)] \\
&\quad - R(x,t+\Delta t)| \quad .
\end{aligned} \tag{2.25}$$

Expanding each of the terms involving Δt or Δx in a Taylor series about $x-t$ for f and about $x+t$ for g results in

$$\begin{aligned}
&|T(x,t+\Delta t)| = \\
&|f(x-t) - \Delta t f'(x-t) + \frac{(\Delta t)^2}{2} f''(x-t) - \frac{(\Delta t)^3}{3!} f'''(x-t) \\
&+\frac{(\Delta t)^4}{4!} f^{iv}(x-t) - ... \\
&+g(x+t) + \Delta t g'(x+t) + \frac{(\Delta t)^2}{2} g''(x+t) + \frac{(\Delta t)^3}{3!} g'''(x+t) \\
&+\frac{(\Delta t)^4}{4!} g^{iv}(x+t) - ... \\
&-2f(x-t) \\
&-2g(x+t) \\
&+f(x-t) + \Delta t f'(x-t) + \frac{(\Delta t)^2}{2} f''(x-t) + \frac{(\Delta t)^3}{3!} f'''(x-t) \\
&+\frac{(\Delta t)^4}{4!} f^{iv}(x-t) + ...
\end{aligned}$$

$$\begin{aligned}
&+g(x+t)-\Delta t g'(x+t)+\frac{(\Delta t)^2}{2}g''(x+t)-\frac{(\Delta t)^3}{3!}g'''(x+t)\\
&+\frac{(\Delta t)^4}{4!}g^{iv}(x-t)-\ldots\\
&-\frac{(\Delta t)^2}{(\Delta x)^2}f(x-t)-\frac{(\Delta t)^2}{(\Delta x)}f'(x-t)-\frac{(\Delta t)^2}{2}f''(x-t)-\frac{(\Delta t)^2(\Delta x)}{3!}f'''(x-t)\\
&-\frac{(\Delta t)^2(\Delta x)^2}{4!}f^{iv}(x-t)-\ldots\\
&-\frac{(\Delta t)^2}{(\Delta x)^2}g(x+t)-\frac{(\Delta t)^2}{(\Delta x)}g'(x+t)-\frac{(\Delta t)^2}{2}g''(x+t)-\frac{(\Delta t)^2(\Delta x)}{3!}g'''(x+t)\\
&-\frac{(\Delta t)^2(\Delta x)^2}{4!}g^{iv}(x+t)-\ldots\\
&+2\frac{(\Delta t)^2}{(\Delta x)^2}f(x-t)\\
&+2\frac{(\Delta t)^2}{(\Delta x)^2}g(x+t)\\
&-\frac{(\Delta t)^2}{(\Delta x)^2}f(x-t)+\frac{(\Delta t)^2}{(\Delta x)}f'(x-t)-\frac{(\Delta t)^2}{2}f''(x-t)+\frac{(\Delta t)^2(\Delta x)}{3!}f'''(x-t)\\
&-\frac{(\Delta t)^2(\Delta x)^2}{4!}f^{iv}(x-t)+\ldots\\
&-\frac{(\Delta t)^2}{(\Delta x)^2}g(x+t)+\frac{(\Delta t)^2}{(\Delta x)}g'(x+t)-\frac{(\Delta t)^2}{2}g''(x+t)+\frac{(\Delta t)^2(\Delta x)}{3!}g'''(x+t)\\
&-\frac{(\Delta t)^2(\Delta x)^2}{4!}g^{iv}(x+t)+\ldots\quad -R(x,t+\Delta t)|\quad .
\end{aligned}$$

Cancellation of terms leaves:

$$\begin{aligned}
&|T(x,t+\Delta t)|\leq\\
&|\frac{(\Delta t)^4}{4!}f^{iv}(x-t)|+O(\Delta t)^6+|\frac{(\Delta t)^4}{4!}g^{iv}(x+t)|+O(\Delta t)^6\\
&+|\frac{(\Delta t)^4}{4!}f^{iv}(x-t)|+O(\Delta t)^6+|\frac{(\Delta t)^4}{4!}g^{iv}(x+t)|+O(\Delta t)^6\\
&+|\frac{(\Delta t)^2(\Delta x)^2}{4!}f^{iv}(x-t)|+O((\Delta t)^2(\Delta x)^4)\\
&+|\frac{(\Delta t)^2(\Delta x)^2}{4!}g^{iv}(x+t)|+O((\Delta t)^2(\Delta x)^4)\\
&+|\frac{(\Delta t)^2(\Delta x)^2}{4!}f^{iv}(x-t)|+O((\Delta t)^2(\Delta x)^4)\\
&+|\frac{(\Delta t)^2(\Delta x)^2}{4!}g^{iv}(x+t)|+O((\Delta t)^2(\Delta x)^4)+|R(x,t+\Delta t)|\quad .
\end{aligned}\tag{2.26}$$

Since $f(x) = \frac{1}{2}sin(x)$ and $g(x) = \frac{1}{2}sin(x)$ for the initial conditions used for this example, equation (2.26) becomes:

$$\begin{aligned}|T(x,t+\Delta t)| \leq & \frac{(\Delta t)^2((\Delta t)^2 + (\Delta x)^2)}{12} + 4[O((\Delta t)^6) + O((\Delta t)^2(\Delta x)^4)] \\ & + |R(x,t+\Delta t)| \quad ,\end{aligned}$$

for general Δx and Δt which, using the convention that global error is one order of magnitude larger than local error, will result in a global error of

$$|T(x,t+\Delta t)| \cong O((\Delta t)((\Delta t)^2 + (\Delta x)^2)) + |R(x,t+\Delta t)| \quad .$$

If we let $\Delta t = \Delta x = \Delta$, then equation (2.25) simplifies to:

$$\begin{aligned}&|T(x,t+\Delta t)| = \\ &|f(x-(t+\Delta)) + g(x+(t+\Delta)) - 2f(x-t) - 2g(x+t) \\ &+f(x-(t-\Delta)) + g(x+(t-\Delta)) - f((x+\Delta)-t) - g((x+\Delta)+t) \\ &+2f(x-t) + 2g(x+t) - f((x-\Delta)-t) - g((x-\Delta)+t) - R(x,t+\Delta t)| \qquad (2.27) \\ &=|f(x-(t+\Delta)) + g(x+(t+\Delta)) - f((x-\Delta)-t) - g((x+\Delta)+t) \\ &-R(x,t+\Delta t)| \\ &=|R(x,t+\Delta t)| \quad ,\end{aligned}$$

for any initial conditions used. The Godunov integration technique results in only machine roundoff error if Δt is chosen equal to Δx and if no errors are committed during the startup step. As seen from equations (2.25 and 2.27), this holds true for all initial conditions.

To estimate the total global truncation error, it is necessary to add the error incurred in the startup step to the global error of the Godunov step. First, an estimate for the error resulting from the use of a forward Euler startup step with the initial conditions of equations (2.17) and (2.18) is found:

$$U(x,\Delta t) = U(x,0) + \Delta t\dot{U}(x,0) + T(x,\Delta t) + R(x,\Delta t),$$

$$\text{or} \quad U(x,\Delta t) = sin(\frac{\pi}{2}x) + T(x,\Delta t) + R(x,\Delta t)$$

since $\dot{U}(x,0) = 0$. Solving for the truncation error gives

$$T(x,\Delta t) = U(x,\Delta t) - sin(\frac{\pi}{2}x) - R(x,\Delta t) \quad .$$

The analytic solution for these initial conditions is

$$U(x,\Delta t) = \frac{1}{2}(sin(\frac{\pi}{2}(x+\Delta t)) + sin(\frac{\pi}{2}(x-\Delta t))) \quad .$$

The error committed during a Forward Euler startup step is then:

$$\begin{aligned}
&|T(x,\Delta t)| = \\
&= |\frac{1}{2}[sin(\frac{\pi}{2}(x+\Delta t)) + sin(\frac{\pi}{2}(x-\Delta t))] - sin(\frac{\pi}{2}x) - R(x,\Delta t)| \\
&= |\frac{1}{2}[sin(\frac{\pi}{2}x)cos(\frac{\pi}{2}\Delta t) + sin(\frac{\pi}{2}\Delta t)cos(\frac{\pi}{2}x) \\
&+ sin(\frac{\pi}{2}x)cos(\frac{\pi}{2}\Delta t) - sin(\frac{\pi}{2}\Delta t)cos(\frac{\pi}{2}x)] - sin(\frac{\pi}{2}x) - R(x,\Delta t)| \\
&= |\frac{1}{2}[2sin(\frac{\pi}{2}x)cos(\frac{\pi}{2}\Delta t)] - sin(\frac{\pi}{2}x) - R(x,\Delta t)| \\
&= |sin(\frac{\pi}{2}x)[cos(\frac{\pi}{2}\Delta t) - 1] - R(x,\Delta t)| \\
&\leq 2|sin(\frac{\pi}{2}x)| + |R(x,\Delta t)| \quad .
\end{aligned}$$

When $\Delta t < 0.1$, a more reasonable estimate for the error results :

$$|T(x,\Delta t| < 0.0123|sin(\frac{\pi}{2}x)| + |R(x,\Delta t)| \quad \text{for } \Delta t < 0.1 \quad .$$

The global truncation error when using a Forward Euler startup with the initial conditions of equations (2.17 and 2.18) is then:

$$|T(x,t+\Delta t)| \leq 2|sin(\frac{\pi}{2}x)| + O((\Delta t)((\Delta t)^2 + (\Delta x)^2)) + |R(x,t+\Delta t)| \quad .$$

For $\Delta t \leq 0.1$ and $\Delta x \leq 0.1$,

$$|T(x,t+\Delta t)| \leq 0.0123|sin(\frac{\pi}{2}x)| + O((\Delta t)((\Delta t)^2 + (\Delta x)^2)) + |R(x,t+\Delta t)| \quad .$$

Next, an estimate is found for the error committed when a Taylor Series Expansion is used for the startup step.

$$U(x,\Delta t) = U(x,0) + \Delta t\dot{U}(x,0) + \frac{(\Delta t)^2}{2}\ddot{U}(x,0) + T(x,\Delta t) + R(x,\Delta t) \quad ,$$

but $\dot{U}(x,0) = 0$, so

$$\begin{aligned} U(x,\Delta t) =& U(x,0) + \frac{(\Delta t)^2}{2}\ddot{U}(x,0) + T(x,\Delta t) + R(x,\Delta t) \qquad (2.28a) \\ =& U(x,0) + \frac{(\Delta t)^2}{2(\Delta x)^2}[U(x+\Delta x,0) - 2U(x,0) + U(x-\Delta x,0)] \\ &+T(x,\Delta t) + R(x,\Delta t) \quad . \qquad (2.28b) \end{aligned}$$

Solving (2.28b) for the truncation error gives the expression

$$\begin{aligned} T(x,\Delta t) = U(x,\Delta t) - U(x,0) - \frac{(\Delta t)^2}{2(\Delta x)^2}[U(x+\Delta x,0) \\ - 2U(x,0) + U(x-\Delta x,0)] - R(x,\Delta t) \quad . \end{aligned}$$

Substituting the exact solution,

$$U(x,\Delta t) = \frac{1}{2}(sin(\frac{\pi}{2}(x+\Delta t)) + sin(\frac{\pi}{2}(x-\Delta t))) \quad ,$$

into the expression for the truncation error gives

$$\begin{aligned}
&|T(x,\Delta t)| = \\
&|\frac{1}{2}sin(\frac{\pi}{2}(x+\Delta t)) + \frac{1}{2}sin(\frac{\pi}{2}(x-\Delta t)) - sin(\frac{\pi}{2}x) \\
&-\frac{(\Delta t)^2}{2(\Delta x)^2}[sin(\frac{\pi}{2}(x+\Delta x)) - 2sin(\frac{\pi}{2}x) + sin(\frac{\pi}{2}(x-\Delta x))] - R(x,\Delta t)| \\
&=|\frac{1}{2}sin(\frac{\pi}{2}x)cos(\frac{\pi}{2}\Delta t) + \frac{1}{2}cos(\frac{\pi}{2}x)sin(\frac{\pi}{2}\Delta t) \\
&+\frac{1}{2}sin(\frac{\pi}{2}x)cos(\frac{\pi}{2}\Delta t) - \frac{1}{2}cos(\frac{\pi}{2}x)sin(\frac{\pi}{2}\Delta t) - sin(\frac{\pi}{2}x) \\
&-\frac{(\Delta t)^2}{2(\Delta x)^2}[sin(\frac{\pi}{2}x)cos(\frac{\pi}{2}\Delta x) + cos(\frac{\pi}{2}x)sin(\frac{\pi}{2}\Delta x) \\
&-2sin(\frac{\pi}{2}x) + sin(\frac{\pi}{2}x)cos(\frac{\pi}{2}\Delta x) - cos(\frac{\pi}{2}x)sin(\frac{\pi}{2}\Delta x)] - R(x,\Delta t)| \\
&=|sin(\frac{\pi}{2}x)cos(\frac{\pi}{2}\Delta t) - sin(\frac{\pi}{2}x) - \frac{(\Delta t)^2}{(\Delta x)^2}[sin(\frac{\pi}{2}x)cos(\frac{\pi}{2}\Delta x) - 2sin(\frac{\pi}{2}x)] \\
&-R(x,\Delta t)| \\
&=|sin(\frac{\pi}{2}x)[cos(\frac{\pi}{2}\Delta t) - 1] - \frac{(\Delta t)^2}{(\Delta x)^2}[sin(\frac{\pi}{2}x)(cos(\frac{\pi}{2}\Delta x) - 1)] - R(x,\Delta t)| \\
&\leq|sin(\frac{\pi}{2}x)[cos(\frac{\pi}{2}\Delta t) - 1]| + |\frac{(\Delta t)^2}{(\Delta x)^2}[sin(\frac{\pi}{2}x)(cos(\frac{\pi}{2}\Delta x) - 1)]| + |R(x,\Delta t)| \\
&\leq|2sin(\frac{\pi}{2}x)(1 + \frac{(\Delta t)^2}{(\Delta x)^2})| + |R(x,\Delta t)| \quad .
\end{aligned}$$

The global truncation error resulting from a Taylor Series Expansion startup and then a Godunov step is then

$$\begin{aligned}
|T(x,t+\Delta t))| \leq &|2sin(\frac{\pi}{2}x)(1 + \frac{(\Delta t)^2}{(\Delta x)^2})| \\
&+O((\Delta t)((\Delta t^2) + (\Delta x)^2)) + |R(x,t+\Delta t)| \quad .
\end{aligned}$$

For $\Delta t \leq 0.1$ and $\Delta x \leq 0.1$, the global error becomes

$$\begin{aligned}
|T(x,t+\Delta t))| \leq &|0.0246sin(\frac{\pi}{2}x)| \\
&+O((\Delta t)((\Delta t^2) + (\Delta x)^2)) + |R(x,t+\Delta t)| \quad .
\end{aligned}$$

When $\Delta t = \Delta x = \Delta$, the error due to the Taylor Series Expansion startup step becomes

$$|T(x,\Delta t)| = |sin(\frac{\pi}{2}x)[cos(\frac{\pi}{2}\Delta) - 1] - \frac{\Delta^2}{\Delta^2}[sin(\frac{\pi}{2}x)[cos(\frac{\pi}{2}\Delta) - 1] - R(x,\Delta t)|$$

$$\text{or} \quad |T(x,\Delta t)| = |R(x,\Delta t)|$$

Thus, the global error for the wave equation,

$$\frac{\partial^2 U}{\partial t^2} = \frac{\partial^2 U}{\partial x^2} \quad ,$$

with the initial conditions of (2.17) and (2.18), with Δt chosen to equal Δx, when solved using a Taylor Series Expansion for the startup step and a Godunov step thereafter, and when the x-axis is discretized using a first order centered difference scheme, is machine roundoff error only.

Floating Point Operations Count

Each Godunov step , $\underline{U}_{n+1} = 2\underline{U}_n - \underline{U}_{n-1} + h^2\ddot{\underline{U}}_n$, when applied to the wave equation which has been discretized in space using a first order centered difference scheme with step size $\Delta x = 0.1$, results in 4 arithmetic operations on vectors of length 10 if h^2 is calculated ahead of time. Each second derivative evaluation needed for the Godunov step takes 39 floating point operations if $\frac{1}{(\Delta x)^2}$ is calculated ahead of time. This gives a total of 79 floating point operations per step required for the Godunov solution. This can be contrasted with the Adams-Bashforth third order solution which takes 7 floating point operations on vectors of length 20 if $\frac{h}{12}$ is calculated ahead of time plus 39 floating point operations for the first derivative evaluations or 179 floating point operations per step.

SECOND EXAMPLE: A MECHANICAL MODEL OF THE HUMAN BODY

$$\frac{d^2\underline{x}(t)}{dt^2} = -\omega^2\underline{x}(t) - C\frac{d\underline{x}(t)}{dt} \tag{2.29}$$

The second set of equations used to test the Godunov integration method results from a passive mechanical system. The particular mechanical system used is a model of the human body designed to give information about how the cervical vertebrae are affected by a bumpy car ride [6].

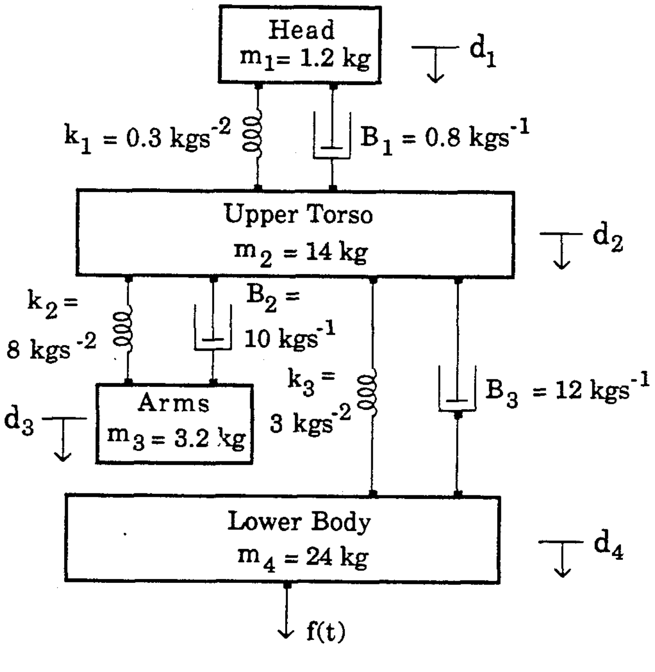


Figure 2.2 Human Body Model

For the second order state equations let

$$x_1 = d_1$$

$$x_2 = d_2$$

$$x_3 = d_3$$

$$x_4 = d_4$$

$$y = x_1 - x_2$$

$$\text{and} \quad u = f \quad .$$

Using the equations:

$$\begin{aligned} F_{mass} &= ma = m\frac{d^2x}{dt^2} = m\ddot{x} \\ F_{damper} &= bv = b\frac{dx}{dt} = b\dot{x} \\ F_{spring} &= kx \quad , \end{aligned}$$

the following set of second order differential equations results:

$$\begin{aligned} \ddot{x}_1 = \frac{F_1}{m_1} =&[k_1(x_2 - x_1) + b_1(\dot{x}_2 - \dot{x}_1)]/m_1 \\ \ddot{x}_2 = \frac{F_2}{m_2} =&[k_2(x_3 - x_2) + b_2(\dot{x}_3 - \dot{x}_2) + k_3(x_4 - x_2) \\ &+ b_3(\dot{x}_4 - \dot{x}_2) - k_1(x_2 - x_1) - b_1(\dot{x}_2 - \dot{x}_1)]/m_2 \\ \ddot{x}_3 = \frac{F_3}{m_3} =&[-k_2(x_3 - x_2) - b_2(\dot{x}_3 - \dot{x}_2)]/m_3 \\ \ddot{x}_4 = \frac{F_4}{m_4} =&[-k_3(x_4 - x_2) - b_3(\dot{x}_4 - \dot{x}_2) + u]/m_4 \quad . \end{aligned} \tag{2.30}$$

Using matrix notation:

$$\begin{aligned} \underline{\ddot{x}}(t) = &-\begin{pmatrix} \frac{k_1}{m_1} & \frac{-k_1}{m_1} & 0 & 0 \\ \frac{-k_1}{m_2} & \frac{k_1+k_2+k_3}{m_2} & \frac{-k_2}{m_2} & \frac{-k_3}{m_2} \\ 0 & \frac{-k_2}{m_3} & \frac{k_2}{m_3} & 0 \\ 0 & \frac{-k_3}{m_4} & 0 & \frac{k_3}{m_4} \end{pmatrix} \underline{x}(t) \\ &-\begin{pmatrix} \frac{b_1}{m_1} & \frac{-b_1}{m_1} & 0 & 0 \\ \frac{-b_1}{m_2} & \frac{b_1+b_2+b_3}{m_2} & \frac{-b_2}{m_2} & \frac{-b_3}{m_2} \\ 0 & \frac{-b_2}{m_3} & \frac{b_2}{m_3} & 0 \\ 0 & \frac{-b_3}{m_4} & 0 & \frac{b_3}{m_4} \end{pmatrix} \underline{\dot{x}}(t) + \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{m_4} \end{pmatrix} u(t) \\ y(t) = &\begin{pmatrix} 1 & -1 & 0 & 0 \end{pmatrix} \underline{x}(t) \quad . \end{aligned}$$

The system can also be set up as a set of first order differential equations by letting

$$x_1 = d_1$$

$$x_2 = \dot{d}_1$$

$$x_3 = d_2$$

$$x_4 = \dot{d}_2$$

$$x_5 = d_3$$

$$x_6 = \dot{d}_3$$

$$x_7 = d_4$$

$$x_8 = \dot{d}_4$$

$$y = x_1 - x_3$$

$$u = f \quad .$$

This results in the following equations:

$$\dot{x}_1 = \dot{d}_1 = x_2$$

$$\dot{x}_2 = \ddot{d}_1 = [k_1(x_3 - x_1) + b_1(x_4 - x_2)]/m_1$$

$$\dot{x}_3 = \dot{d}_2 = x_4$$

$$\dot{x}_4 = \ddot{d}_2 = [k_2(x_5 - x_3) + b_2(x_6 - x_4) + k_3(x_7 - x_3) \\ + b_3(x_8 - x_4) - k_1(x_3 - x_1) - b_1(x_4 - x_2)]/m_2$$

$$\dot{x}_5 = \dot{d}_3 = x_6$$

$$\dot{x}_6 = \ddot{d}_3 = [-k_2(x_5 - x_3) - b_2(x_6 - x_4)]/m_3$$

$$\dot{x}_7 = \dot{d}_4 = x_8$$

$$\dot{x}_8 = \ddot{d}_4 = [-k_3(x_7 - x_3) - b_3(x_8 - x_4) + u]/m_4 \quad ,$$

or, using matrix notation,

$$\underline{\dot{x}}(t) = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{-k_1}{m_1} & \frac{-b_1}{m_1} & \frac{k_1}{m_1} & \frac{b_1}{m_1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{k_1}{m_2} & \frac{b_1}{m_2} & \frac{-k_1-k_2-k_3}{m_2} & \frac{-b_1-b_2-b_3}{m_2} & \frac{k_2}{m_2} & \frac{b_2}{m_2} & \frac{k_3}{m_2} & \frac{b_3}{m_2} \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{k_2}{m_3} & \frac{b_2}{m_3} & \frac{-k_2}{m_3} & \frac{-b_2}{m_3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{k_3}{m_4} & \frac{b_3}{m_4} & 0 & 0 & \frac{-k_3}{m_4} & \frac{-b_3}{m_4} \end{pmatrix} \underline{x}(t)$$

$$+ \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{m_4} \end{pmatrix} u(t) \quad . \tag{2.31}$$

Stability Analysis for Godunov Integration of the Human Body Model ($\underline{\ddot{x}}(t) = -\omega^2 \underline{x}(t) - C\underline{\dot{x}}(t)$)

First a stability analysis will be attempted for a scalar system by using the substitution $z(t) = \dot{x}(t)$ which will result in two equations:

$$\begin{aligned} \ddot{x}(t) &= -\omega^2 x(t) - cz(t) \\ \dot{z}(t) &= -\omega^2 x(t) - cz(t) \quad . \end{aligned} \tag{2.32}$$

Using a Godunov step to extrapolate x_{n+1}, we get the equation

$$x_{n+1} - 2x_n + x_{n-1} = h^2 \ddot{x}_n = -\omega^2 h^2 x_n - ch^2 z_n \quad , \tag{2.33}$$

and using an Adams-Bashforth 3rd order step to extrapolate z_{n+1}, we get the equation

$$\begin{aligned} z_{n+1} - z_n =& \frac{h}{12}[23(-\omega^2 x_n - cz_n) \\ & -16(-\omega^2 x_{n-1} - cz_{n-1}) + 5(-\omega^2 x_{n-2} - cz_{n-2})] \quad . \end{aligned} \tag{2.34}$$

Assuming solutions $x_n = r^n x_0$ and $z_n = r^n z_0$ and substituting these solutions in equations (2.33) and (2.34),

$$\begin{aligned}(r^{n+1} - 2r^n + r^{n-1})x_0 &= -\,\omega^2 h^2 r^n x_0 - ch^2 r^n z_0 \quad , \\ (r^{n+1} - r^n)z_0 &= \frac{h}{12}[-23\omega^2 r^n + 16\omega^2 r^{n-1} - 5\omega^2 r^{n-2}]x_0 \\ &\quad + \frac{h}{12}[-23cr^n + 16cr^{n-1} - 5cr^{n-2}]z_0 \quad .\end{aligned}$$

Rearranging both equations results in,

$$\begin{pmatrix} (r^2 + (h^2\omega^2 - 2)r + 1) & (h^2 cr) \\ (\frac{23}{12}h\omega^2 r^2 - \frac{16}{12}h\omega^2 r + \frac{5}{12}h\omega^2) & (r^3 + (\frac{23}{12}hc - 1)r^2 - \frac{16}{12}hcr + \frac{5}{12}hc) \end{pmatrix} \begin{pmatrix} x_0 \\ z_0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad . \tag{2.35}$$

Although initial conditions of $x_0 = 0$ and $z_0 = 0$ are used when this problem is solved with an input, this stability analysis is being done for equations (2.32), which have initial conditions but no input, so nonzero x_0 and z_0 must be used to give a nontrivial solution for r. For nonzero $\underline{\xi}$, equations of the form $A\underline{\xi} = 0$ have a solution if $|A| = 0$, so equation (2.35) has a solution if

$$\begin{aligned}&[r^2 + (h^2\omega^2 - 2)r + 1][r^3 + (\frac{23}{12}hc - 1)r^2 - \frac{16}{12}hcr + \frac{5}{12}hc] \\ &- [\frac{23}{12}h\omega^2 r^2 - \frac{16}{12}h\omega^2 r + \frac{5}{12}h\omega^2][h^2 cr] = 0 \qquad (2.36a) \\ \text{or} \quad &r^5 + r^4(h^2\omega^2 + \frac{23}{12}hc - 3) + r^3(-h^2\omega^2 - \frac{31}{6}hc + 3) \\ &+ r^2(5hc - 1) + r(-\frac{13}{6}hc) + \frac{5}{12}hc = 0 \quad . \qquad (2.36b)\end{aligned}$$

The roots of equation (2.36b) are easily found numerically by plugging the c, h and ω^2 values from the scalar system of equations (2.32) into equation (2.36b). In an attempt to determine the stability of the vector system actually used in this example, however, another approach was tried. If equations (2.33) and (2.34) are rearranged in the form

$$\underline{x}_{n+1} = A\underline{x}_n \quad , \tag{2.37a}$$

$$\text{or} \quad \underline{x}_n = A^n\underline{x}_0 \quad , \tag{2.37b}$$

then the largest eigenvalue of A must have a magnitude of less than one in order for the method to be stable. This is accomplished by solving the vector form of equations (2.33) and (2.34) for $\underline{x}_{n+1}$ and $\underline{z}_{n+1}$ respectively which results in two equations for $\underline{x}$ and $\underline{z}$ at time step $n+1$ each in terms of $\underline{x}$ and $\underline{z}$ at time steps n, $n-1$, and $n-2$:

$$\begin{aligned}
\underline{x}_{n+1} =& h^2\ddot{\underline{x}}_n + 2\underline{x}_n - \underline{x}_{n-1} \\
=& h^2(-\omega^2\underline{x}_n - C\underline{z}_n) + 2\underline{x}_n - \underline{x}_{n-1} \\
=& (2I - h^2\omega^2)\underline{x}_n - h^2C\underline{z}_n - \underline{x}_{n-1} && (2.38) \\
\underline{z}_{n+1} =& \frac{-23}{12}h\omega^2\underline{x}_n + \frac{4}{3}h\omega^2\underline{x}_{n-1} - \frac{5}{12}h\omega^2\underline{x}_{n-2} \\
&+ (I - \frac{23}{12}hC)\underline{z}_n + \frac{4}{3}hC\underline{z}_{n-1} - \frac{5}{12}hC\underline{z}_{n-2} \quad . && (2.29)
\end{aligned}$$

Equations (2.38) and (2.39) can be brought into the form of equation (2.37b) by making the following substitutions:

$$\begin{aligned}
\underline{y}_n &= \underline{x}_{n-1} \\
\underline{v}_n &= \underline{x}_{n-2} \\
\underline{w}_n &= \underline{z}_{n-1} \\
\underline{u}_n &= \underline{z}_{n-2} \quad .
\end{aligned}$$

These substitutions result in the equations

$$\underline{y}_{n+1} = \underline{x}_n$$

$$\underline{v}_{n+1} = \underline{x}_{n-1} = \underline{y}_n$$

$$\underline{w}_{n+1} = \underline{z}_n$$

$$\underline{u}_{n+1} = \underline{z}_{n-1} = \underline{w}_n$$

$$\underline{x}_{n+1} = (2I - h^2\omega^2)\underline{x}_n - h^2C\underline{z}_n - \underline{y}_n$$

$$\underline{z}_{n+1} = \frac{-23}{12}h\omega^2\underline{x}_n + \frac{4}{3}h\omega^2\underline{y}_n - \frac{5}{12}h\omega^2\underline{v}_n + (I - \frac{23}{12}hC)\underline{z}_n + \frac{4}{3}hC\underline{w}_n - \frac{5}{12}hC\underline{u}_n$$

which, if put into the form of equation (2.37b), will result in:

$$\begin{pmatrix} \underline{y} \\ \underline{v} \\ \underline{w} \\ \underline{u} \\ \underline{x} \\ \underline{z} \end{pmatrix}_n = \begin{pmatrix} 0 & 0 & 0 & 0 & I & 0 \\ I & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I \\ 0 & 0 & I & 0 & 0 & 0 \\ -I & 0 & 0 & 0 & 2I - h^2\omega^2 & -h^2C \\ \frac{4}{3}h\omega^2 & -\frac{5}{12}h\omega^2 & \frac{4}{3}hC & -\frac{5}{12}hC & -\frac{23}{12}h\omega^2 & I - \frac{23}{12}hC \end{pmatrix}^n \begin{pmatrix} \underline{y} \\ \underline{v} \\ \underline{w} \\ \underline{u} \\ \underline{x} \\ \underline{z} \end{pmatrix}_0$$

ω^2 and C come from equation (2.29). The following table shows the relationship between h and the largest eigenvalue of this equation:

h	$\lvert\lambda\rvert_{max}$
.01	1.000000000000012
.1	1.000000006649771
.11	1.000000002754222
.12	1.000000008708826
.13	1.000000005795538
.14	1.059315692193166
.15	1.132070446910589

Table 2.1 Magnitudes of Eigenvalues of Discretized Mechanical System

When the largest eigenvalue of the A matrix is just slightly greater than one, the system has proven to be stable experimentally, so some roundoff error must be causing the eigenvalue to be greater than one. The eigenvalue of 1.05 at $h = .14$, however, is too large to be explained away by roundoff error so this must be the maximum stepsize allowed for stability.

Alternative Stability Analysis

If a leap-frog step is used for the extrapolation of the first derivative $\dot{x}$, an analytic stability domain can be found for the problem

$$\ddot{x}(t) = -\omega^2 x(t) - cz(t) \tag{2.40}$$

$$\text{and} \quad \dot{z}(t) = -\omega^2 x(t) - cz(t) \quad . \tag{2.41}$$

Using a Godunov step and a leap-frog step on equations (2.40) and (2.41) respectively, we get the equations

$$\ddot{x}_n = \frac{x_{n+1} - 2x_n + x_{n-1}}{h^2} \tag{2.42}$$

$$\dot{x}_n = z_n = \frac{x_{n+1} - x_{n-1}}{2h} \quad . \tag{2.43}$$

Substituting equations (2.42) and (2.43) into equation (2.40) results in

$$[\frac{x_{n+1} - 2x_n + x_{n-1}}{h^2}] + c[\frac{x_{n+1} - x_{n-1}}{2h}] + \omega^2 x_n = 0 \quad . \tag{2.44}$$

If the solution $x_n = r^n$ is assumed for equation (2.44), the quadratic

$$[1 + \frac{ch}{2}]r^2 + [h^2\omega^2 - 2]r + [1 - \frac{ch}{2}] = 0 \tag{2.45}$$

results. The solution to equation (2.45) is

$$r = \frac{1}{2+ch}[2 - h^2\omega^2 \pm i\omega h\sqrt{4 - h^2\omega^2 - \frac{c^2}{\omega^2}}]$$

$$\text{or} \quad |r|^2 = [\frac{1}{2+ch}]^2[(2 - h^2\omega^2)^2 + \omega^2h^2(4 - h^2\omega^2 - \frac{c^2}{\omega^2})]$$

$$= \frac{4 - c^2h^2}{(2+ch)^2} = \frac{(2+ch)(2-ch)}{(2+ch)^2} = \frac{2-ch}{2+ch} < 1 \quad \text{if} \quad c > 0 \quad \text{and if} \quad h > 0 \quad .$$

This calculation for r is valid only if

$$4 - h^2\omega^2 - \frac{c^2}{\omega^2} > 0$$

$$\text{or} \quad h < \frac{1}{\omega}\sqrt{4 - \frac{c^2}{\omega^2}} \quad .$$

Error Analysis for Godunov Integration

Rewriting equations (2.38) and (2.39) gives

$$\underline{x}(t + \Delta t) = 2\underline{x}(t) - \underline{x}(t - \Delta t) + (\Delta t)^2\underline{\ddot{x}}(t) + \underline{T}(t + \Delta t) + \underline{R}(t + \Delta t) \qquad (2.46)$$

$$\begin{aligned} \underline{z}(t + \Delta t) =& \underline{z}(t) + \frac{23}{12}(\Delta t)\underline{\dot{z}}(t) - \frac{4}{3}(\Delta t)\underline{\dot{z}}(t - \Delta t) \\ &+ \frac{5}{12}(\Delta t)\underline{\dot{z}}(t - 2\Delta t) + \underline{T}(t + \Delta t) + \underline{R}(t + \Delta t) \quad , \end{aligned} \qquad (2.47)$$

where $\underline{T}$ and $\underline{R}$ denote the truncation error and the roundoff error. An expression for the local truncation error incurred in the Godunov step can be found by solving equations (2.46) and (2.47) for $\underline{T}(t + \Delta t)$ and using Taylor series expansions on the result:

$$\begin{aligned} &|\underline{T}((t + \Delta t)| = \\ &|\underline{x}(t) + (\Delta t)\underline{\dot{x}}(t) + \frac{(\Delta t)^2}{2}\underline{\ddot{x}}(t) + \frac{(\Delta t)^3}{6}\underline{x}^{(iii)}(t) + \frac{(\Delta t)^4}{24}\underline{x}^{(iv)}(t) + ... \\ &- 2\underline{x}(t) \\ &+ \underline{x}(t) - (\Delta t)\underline{\dot{x}}(t) + \frac{(\Delta t)^2}{2}\underline{\ddot{x}}(t) - \frac{(\Delta t)^3}{6}\underline{x}^{(iii)}(t) + \frac{(\Delta t)^4}{24}\underline{x}^{(iv)}(t) - ... \\ &- (\Delta t)^2\underline{\ddot{x}}(t) - \underline{R}(t + \Delta t)| \\ &\leq |\frac{(\Delta t)^4}{12}\underline{x}^{(iv)}(t)| + O((\Delta t)^6) + |\underline{R}(t + \Delta t)| \quad . \end{aligned}$$

The global error is then

$$|\underline{T}(t+\Delta t))| \leq O((\Delta t)^3) + |\underline{R}(t+\Delta t)| \quad .$$

An expression for the error incurred in each Adams-Bashforth 3rd order step on $\underline{z}(t)$ is derived in a similar manner:

$$\begin{aligned}
&|\underline{T}((t+\Delta t)| = \\
&|\underline{z}(t) + (\Delta t)\underline{\dot{z}}(t) + \frac{(\Delta t)^2}{2}\underline{\ddot{z}}(t) + \frac{(\Delta t)^3}{6}\underline{z}^{(iii)}(t) + \frac{(\Delta t)^4}{24}\underline{z}^{(iv)}(t) + ... \\
&- \underline{z}(t) - \frac{23}{12}(\Delta t)\underline{\dot{z}}(t) \\
&+ \frac{4}{3}(\Delta t)\underline{\dot{z}}(t) - \frac{4}{3}(\Delta t)^2\underline{\ddot{z}}(t) + \frac{2}{3}(\Delta t)^3\underline{z}^{(iii)}(t) - \frac{2}{9}(\Delta t)^4\underline{z}^{(iv)}(t) + ... \\
&- \frac{5}{12}(\Delta t)\underline{\dot{z}}(t) + \frac{5}{6}(\Delta t)^2\underline{\ddot{z}}(t) - \frac{5}{6}(\Delta t)^3\underline{z}^{(iii)}(t) + \frac{5}{9}(\Delta t)^4\underline{z}^{(iv)}(t) + ... - \underline{R}(t+\Delta t) \\
&\leq |\frac{3}{8}(\Delta t)^4\underline{z}^{(iv)}(t)| + O((\Delta t)^5) + |\underline{R}(t+\Delta t)| \\
&= |\frac{3}{8}(\Delta t)^4\underline{x}^{(v)}(t)| + O((\Delta t)^5) + |\underline{R}(t+\Delta t)| \quad ,
\end{aligned}$$

which also gives a global truncation error of $O((\Delta t)^3) + |\underline{R}(t+\Delta t)|$.

Floating Point Operations Count

The Godunov method, $\underline{x}_{n+1} = 2\underline{x}_n - \underline{x}_{n-1} + h^2\underline{\ddot{x}}_n$, when used for the human body model, results in 4 arithmetic operations on vectors of length 4 if h^2 is calculated ahead of time, or 16 floating point operations per step. The Adams Bashforth 3rd order method, $\underline{z}_{n+1} = \underline{z}_n + \frac{h}{12}[23\underline{\dot{z}}_n - 16\underline{\dot{z}}_{n-1} + 5\underline{\dot{z}}_{n-2}]$ where $\underline{z}_n = \underline{\dot{x}}_n$, when used on the human body model, results in 7 arithmetic operations on vectors of length 4 if $\frac{h}{12}$ is calculated ahead of time, or 28 floating point operations per step. Each second derivative calculation for the human body model, which needs to be done only once per step, results in 38 floating point operations. It can be seen by summing floating point operations that the Godunov method, when applied in conjunction with the Adams-Bashforth 3rd order method to solve the human body model, results in 82 floating point operations per step. This can be contrasted with the use of the Adams-Bashforth 3rd order method when used to solve equation (2.31), which results in 94 floating point operations per step. Thus, the Godunov method should take 12 fewer floating point operations per step than the Adams-Bashforth 3rd order method when used on this particular system.

THIRD EXAMPLE: A 7th ORDER ELECTRICAL SYSTEM

The most general form of the equation to be solved using the Godunov method is

$$\begin{aligned}\frac{d^2\underline{x}_1(t)}{dt^2} &= \underline{f}_1(\underline{x}_1(t), \frac{d\underline{x}_1(t)}{dt}, \underline{x}_2(t), \underline{u}(t), t)\\ \frac{d\underline{x}_2(t)}{dt} &= \underline{f}_2(\underline{x}_1(t), \frac{d\underline{x}_1(t)}{dt}, \underline{x}_2(t), \underline{u}(t), t)\\ \underline{y}(t) &= \underline{g}(\underline{x}_1(t), \frac{d\underline{x}_1(t)}{dt}, \underline{x}_2(t), \underline{u}(t), t)\end{aligned}$$

In an attempt to solve a system which had equations closer to this general form, and also to test the Godunov method on a system from the electrical engineering world, a passive electrical network was used for the third example (Figure 2.3). This system resulted in the set of equations

$$\begin{aligned}\frac{d^2\underline{x}_1(t)}{dt^2} &= A\underline{x}_1(t) + B\frac{d\underline{x}_1(t)}{dt} + \underline{c}x_2(t) + \underline{d}\frac{du(t)}{dt}\\ \frac{dx_2(t)}{dt} &= \underline{e}'\underline{x}_1(t) + fx_2(t)\\ y(t) &= x_2(t) \quad .\end{aligned} \tag{2.48}$$

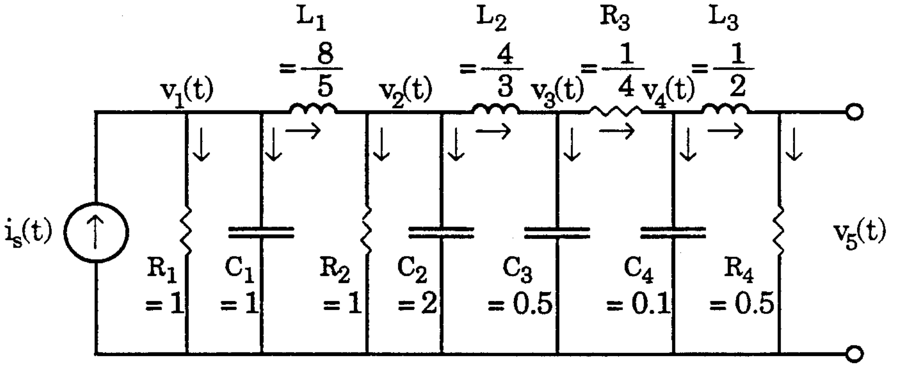


Figure 2.3 Passive Electrical System

The first step in developing the differential equations to be solved with the Godunov method is to sum currents at each node, resulting in five integro-differential equations:

$$
\begin{aligned}
&\frac{v_1(t)}{R_1} + C_1\frac{dv_1(t)}{dt} + \frac{1}{L_1}\int (v_1(t) - v_2(t))\,dt = i_s(t) \\
&\frac{v_2(t)}{R_2} + C_2\frac{dv_2(t)}{dt} + \frac{1}{L_2}\int (v_2(t) - v_3(t))\,dt + \frac{1}{L_1}\int (v_2(t) - v_1(t))\,dt = 0 \\
&\frac{v_3(t) - v_4(t)}{R_3} + C_3\frac{dv_3(t)}{dt} + \frac{1}{L_2}\int (v_3(t) - v_2(t))\,dt = 0 \\
&C_4\frac{dv_4(t)}{dt} + \frac{1}{L_3}\int (v_4(t) - v_5(t))\,dt + \frac{v_4(t) - v_3(t)}{R_3} = 0 \\
&\frac{1}{L_3}\int (v_5(t) - v_4(t))dt + \frac{v_5(t)}{R_4} = 0 \quad .
\end{aligned}
\tag{2.49}
$$

Since the differential equation solver needs a set of differential equations without integrals, we can differentiate equations (2.49) to obtain the result:

$$
\begin{aligned}
&C_1\ddot{v}_1(t) + \frac{\dot{v}_1(t)}{R_1} + \frac{v_1(t)}{L_1} - \frac{v_2(t)}{L_1} = \frac{di_s(t)}{dt} \\
&C_2\ddot{v}_2(t) + \frac{\dot{v}_2(t)}{R_2} + \frac{v_2(t)}{L_2} + \frac{v_2(t)}{L_1} - \frac{v_3(t)}{L_2} - \frac{v_1(t)}{L_1} = 0 \\
&C_3\ddot{v}_3(t) + \frac{\dot{v}_3(t)}{R_3} - \frac{\dot{v}_4(t)}{R_3} + \frac{v_3(t)}{L_2} - \frac{v_2(t)}{L_2} = 0 \\
&C_4\ddot{v}_4(t) + \frac{\dot{v}_4(t)}{R_3} - \frac{\dot{v}_3(t)}{R_3} + \frac{v_4(t)}{L_3} - \frac{v_5(t)}{L_3} = 0 \\
&\frac{\dot{v}_5(t)}{R_4} + \frac{v_5(t)}{L_3} - \frac{v_4(t)}{L_3} = 0 \quad .
\end{aligned}
\tag{2.50}
$$

Solving equations (2.50) for the highest derivative results in:

$$
\begin{aligned}
\ddot{v}_1(t) &= -\frac{1}{R_1C_1}\dot{v}_1(t) - \frac{1}{L_1C_1}v_1(t) + \frac{1}{L_1C_1}v_2(t) + \frac{1}{C_1}\frac{di_s(t)}{dt} \\
\ddot{v}_2(t) &= -\frac{1}{R_2C_2}\dot{v}_2(t) + \frac{1}{L_1C_2}v_1(t) - \frac{L_1 + L_2}{C_2L_1L_2}v_2(t) + \frac{1}{L_2C_2}v_3(t) \\
\ddot{v}_3(t) &= -\frac{1}{C_3R_3}\dot{v}_3(t) + \frac{1}{C_3R_3}\dot{v}_4(t) + \frac{1}{L_2C_3}v_2(t) - \frac{1}{L_2C_3}v_3(t) \\
\ddot{v}_4(t) &= \frac{1}{C_4R_3}\dot{v}_3(t) - \frac{1}{C_4R_3}\dot{v}_4(t) - \frac{1}{L_3C_4}v_4(t) + \frac{1}{L_3C_4}v_5(t) \\
\dot{v}_5(t) &= \frac{R_4}{L_3}v_4(t) - \frac{R_4}{L_3}v_5(t) \quad .
\end{aligned}
\tag{2.51}
$$

Letting

$$\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} = \underline{x}_1 \quad \text{and} \quad v_5 = x_2 \quad \text{and} \quad i_s = u$$

and writing equations (2.51) in the form of equations (2.48) results in

$$\ddot{\underline{x}}_1(t) = \begin{pmatrix} -\frac{1}{L_1C_1} & \frac{1}{L_1C_1} & 0 & 0 \\ \frac{1}{L_1C_2} & -\frac{(L_1+L_2)}{C_2L_1L_2} & \frac{1}{L_2C_2} & 0 \\ 0 & \frac{1}{L_2C_3} & -\frac{1}{L_2C_3} & 0 \\ 0 & 0 & 0 & -\frac{1}{L_3C_4} \end{pmatrix} \underline{x}_1(t) +$$

$$\begin{pmatrix} -\frac{1}{R_1C_1} & 0 & 0 & 0 \\ 0 & -\frac{1}{R_2C_2} & 0 & 0 \\ 0 & 0 & -\frac{1}{R_3C_3} & \frac{1}{R_3C_3} \\ 0 & 0 & \frac{1}{R_3C_4} & -\frac{1}{R_3C_4} \end{pmatrix} \dot{\underline{x}}_1(t) +$$

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{L_3C_4} \end{pmatrix} x_2(t) + \begin{pmatrix} \frac{1}{C_1} \\ 0 \\ 0 \\ 0 \end{pmatrix} \dot{u}(t)$$

$$\dot{x}_2(t) = \begin{pmatrix} 0 & 0 & 0 & \frac{R_4}{L_3} \end{pmatrix} \underline{x}_1(t) + \left(-\frac{R_4}{L_3} \right) x_2(t)$$

$$y(t) = x_2(t) \quad .$$

The electrical system can also be set up as a system of 1st order equations by separating each second order ODE of equations (2.51) into 2 first order ODEs or by using currents through inductors and voltages on capacitors as state variables.

If the latter method is used, seven 1st order ODEs result.

$$
\begin{aligned}
x_1(t) &= v_{C_1}(t) & \dot{x}_1(t) &= \frac{1}{C_1} i_{C_1}(t) = \frac{-1}{R_1 C_1} x_1(t) - \frac{1}{C_1} x_2(t) + \frac{1}{C_1} u(t) \\
x_2(t) &= i_{L_1}(t) & \dot{x}_2(t) &= \frac{1}{L_1} v_{L_1}(t) = \frac{1}{L_1} x_1(t) - \frac{1}{L_1} x_3(t) \\
x_3(t) &= v_{C_2}(t) & \dot{x}_3(t) &= \frac{1}{C_2} i_{C_2}(t) = \frac{1}{C_2} x_2(t) - \frac{1}{C_2 R_2} x_3(t) - \frac{1}{C_2} x_4(t) \\
x_4(t) &= i_{L_2}(t) & \dot{x}_4(t) &= \frac{1}{L_2} v_{L_2}(t) = \frac{1}{L_2} x_3(t) - \frac{1}{L_2} x_5(t) \\
x_5(t) &= v_{C_3}(t) & \dot{x}_5(t) &= \frac{1}{C_3} i_{C_3}(t) = \frac{1}{C_3} x_4(t) - \frac{1}{C_3 R_3} x_5(t) + \frac{1}{C_3 R_3} x_6(t) \\
x_6(t) &= v_{C_4}(t) & \dot{x}_6(t) &= \frac{1}{C_6} i_{C_6}(t) = \frac{1}{C_4 R_3} x_5(t) - \frac{1}{C_4 R_3} x_6(t) - \frac{1}{C_4} x_7(t) \\
x_7(t) &= i_{L_3}(t) & \dot{x}_7(t) &= \frac{1}{L_3} v_{L_3}(t) = \frac{1}{L_3} x_6(t) - \frac{R_4}{L_3} x_7(t) \quad .
\end{aligned}
\tag{2.52}
$$

Or, equations (2.52) can be written in matrix form:

$$
\begin{aligned}
\underline{\dot{x}}(t) = {} & \begin{pmatrix}
-\frac{1}{R_1 C_1} & -\frac{1}{C_1} & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{L_1} & 0 & -\frac{1}{L_1} & 0 & 0 & 0 & 0 \\
0 & \frac{1}{C_2} & -\frac{1}{C_2 R_2} & -\frac{1}{C_2} & 0 & 0 & 0 \\
0 & 0 & \frac{1}{L_2} & 0 & -\frac{1}{L_2} & 0 & 0 \\
0 & 0 & 0 & \frac{1}{C_3} & -\frac{1}{C_3 R_3} & \frac{1}{C_3 R_3} & 0 \\
0 & 0 & 0 & 0 & \frac{1}{C_4 R_3} & -\frac{1}{C_4 R_3} & -\frac{1}{C_4} \\
0 & 0 & 0 & 0 & 0 & \frac{1}{L_3} & -\frac{R_4}{L_3}
\end{pmatrix} \underline{x}(t) \\
& + \begin{pmatrix} \frac{1}{C_1} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} u(t) \\
y(t) = {} & (0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad R_4)\, \underline{x}(t) \quad .
\end{aligned}
\tag{2.53}
$$

Note about Initial Conditions

It was discovered that, due to the two different methods used to develop the second order equations and the first order equations, care must be taken to ensure that the initial conditions of the two representations are consistent with each other. Relating the state variables of each representation to the circuit diagram of Figure (2.4), it can be seen that for the first order system,

$$\begin{aligned} x_1 &= v_{C_1} \\ x_2 &= i_{L_1} \\ x_3 &= v_{C_2} \\ x_4 &= i_{L_2} \\ x_5 &= v_{C_3} \\ x_6 &= v_{C_4} \\ x_7 &= i_{L_3} \quad , \end{aligned}$$

and for the second order system,

$$\begin{aligned} x_{1_1} &= v_{C_1} \\ x_{1_2} &= v_{C_2} \\ x_{1_3} &= v_{C_3} \\ x_{1_4} &= v_{C_4} \\ x_2 &= R_4 i_{L_3} \quad , \end{aligned}$$

and

$$\begin{aligned} \frac{dx_{1_1}}{dt} &= \frac{dv_{C_1}}{dt} = \frac{-1}{R_1C_1}v_{C_1} - \frac{1}{C_1}i_{L_1} + \frac{1}{C_1}i_s \\ \frac{dx_{1_2}}{dt} &= \frac{dv_{C_2}}{dt} = \frac{1}{C_2}i_{L_1} - \frac{1}{C_2R_2}v_{C_2} - \frac{1}{C_2}i_{L_2} \\ \frac{dx_{1_3}}{dt} &= \frac{dv_{C_3}}{dt} = \frac{1}{C_3}i_{L_2} - \frac{1}{C_3R_3}v_{C_3} + \frac{1}{C_3R_3}v_{C_4} \\ \frac{dx_{1_4}}{dt} &= \frac{dv_{C_4}}{dt} = \frac{1}{C_4R_3}v_{C_3} - \frac{1}{C_4R_3}v_{C_4} - \frac{1}{C_4}i_{L_3} \quad . \end{aligned}$$

Clearly, if the initial conditions of all the states in the first order system are chosen to be 0, then, to be consistent with this, the initial conditions of all the states and their derivatives of the second order system will be zero except possibly $\frac{dx_{1_1}(0)}{dt}$ if $i_s(0) \neq 0$. Since the input chosen for this example was $i_s(t) = 144cos3t$, then $i_s(0) = 144 = \frac{dx_{1_1}(0)}{dt}$, and all other initial conditions for the second order system are zero.

Stability Analysis for Godunov Integration of Electrical System

In a manner similar to that used for the stability analysis of the mechanical system, the first two equations of (2.48) can be written

$$\begin{aligned}
\ddot{\underline{x}}_1(t) &= A\underline{x}_1(t) + B\underline{z}(t) + \underline{c}x_2(t) + \underline{d}\dot{u}(t) \\
\dot{\underline{z}}(t) &= A\underline{x}_1(t) + B\underline{z}(t) + \underline{c}x_2(t) + \underline{d}\dot{u}(t) \\
\dot{x}_2(t) &= \underline{e}'\underline{x}_1(t) + fx_2(t)
\end{aligned}$$

where $\underline{z}(t) = \dot{\underline{x}}_1(t)$. By letting

$$\underline{y}(t) = \begin{pmatrix} \underline{z}(t) \\ x_2(t) \end{pmatrix} \quad ,$$

a set of equations which are similar to those of the human body model results:

$$\ddot{\underline{x}}_1(t) = A\underline{x}_1(t) + (B \quad \underline{c})\,\underline{y}(t) + \underline{d}\dot{u}(t) \tag{2.54a}$$

$$\dot{\underline{y}}(t) = \begin{pmatrix} A \\ \underline{e}' \end{pmatrix} \underline{x}_1(t) + \begin{pmatrix} B & \underline{c} \\ 0 & f \end{pmatrix} \underline{y}(t) + \begin{pmatrix} \underline{d} \\ 0 \end{pmatrix} \dot{u}(t) \quad . \tag{2.54b}$$

If equation (2.54a) is solved using a Godunov step and equation (2.54b) is solved using an Adams-Bashforth 3rd order step, and if the substitutions,

$$\begin{aligned}
\underline{p}_n &= \underline{x}_{1_{n-1}} \\
\underline{v}_n &= \underline{x}_{1_{n-2}} = \underline{p}_{n-1} \\
\underline{w}_n &= \underline{y}_{n-1} \\
\underline{q}_n &= \underline{y}_{n-2} = \underline{w}_{n-1} \quad ,
\end{aligned}$$

are made, a set of equations identical in form to those used for the mechanical model of the human body result:

$$
\begin{aligned}
\underline{v}_{n+1} &= \underline{p}_n \\
\underline{q}_{n+1} &= \underline{w}_n \\
\underline{p}_{n+1} &= \underline{x}_{1_n} \\
\underline{w}_{n+1} &= \underline{y}_n \\
\underline{x}_{1_{n+1}} &= (h^2 A + 2I)\underline{x}_{1_n} - \underline{p}_n + h^2 \begin{pmatrix} B & \underline{c} \end{pmatrix} \underline{y}_n \\
\underline{y}_{n+1} &= \frac{23}{12} h \begin{pmatrix} A \\ \underline{e}' \end{pmatrix} \underline{x}_{1_n} + (\frac{23}{12} h \begin{pmatrix} B & \underline{c} \\ 0 & f \end{pmatrix} + I)\underline{y}_n - \frac{4}{3} h \begin{pmatrix} A \\ \underline{e}' \end{pmatrix} \underline{p}_n \\
& \quad - \frac{4}{3} h \begin{pmatrix} B & \underline{c} \\ 0 & f \end{pmatrix} \underline{w}_n + \frac{5}{12} h \begin{pmatrix} A \\ \underline{e}' \end{pmatrix} \underline{v}_n + \frac{5}{12} h \begin{pmatrix} B & \underline{c} \\ 0 & f \end{pmatrix} \underline{q}_n \quad .
\end{aligned}
\tag{2.55}
$$

If equations (2.55) are put into the form $\underline{x}_{n+1} = A\underline{x}_n$, or $\underline{x}_n = A^n \underline{x}_0$ then the eigenvalues of the A matrix must all have magnitudes of less than one to insure stability of the numerical system. Table (2.2) shows the relationship between h and the largest eigenvalue of the A matrix:

h	$\lvert\lambda\rvert_{max}$
.001	1.000000003131651
.005	1.000000008101029
.01	1.000000000708128
.011	1.000000007505733
.012	1.052082324279837
.013	1.133711220949891

Table 2.2 Magnitudes of Eigenvalues of Discretized Electrical System

Using the same reasoning as for the mechanical system, the largest step size allowed for stability of the electrical system should be $h = .012$.

Error Analysis for Hybrid Integration Method

Using a Godunov step for equation (2.54a) and an Adams-Bashforth 3rd order step for equation (2.54b), the same error equations that were found for the mechanical system result:

$$\begin{aligned}
|\underline{T}_{\underline{x}_1}(t+\Delta t)| = &= |\frac{(\Delta t)^4}{12}\underline{x}_1{}^{(iv)}(t)| + O((\Delta t)^6) + |\underline{R}(t+\Delta t)| \\
&= O((\Delta t)^3) + |\underline{R}(t+\Delta t)| \quad \text{globally,} \\
|\underline{T}_{\underline{y}}(t+\Delta t)| &= |\frac{3}{8}(\Delta t)^4 \underline{y}^{(iv)}(t)| + O((\Delta t)^5) + |\underline{R}(t+\Delta t)| \\
&= O((\Delta t)^3) + |\underline{R}(t+\Delta t)| \quad \text{globally.}
\end{aligned}$$

These errors are not explicitly calculated here due to the complexity of the analytic solution.

Floating Point Operations Count

The Godunov method when used on the 7th order electrical system to extrapolate $\underline{x}_1$ results in 4 arithmetic operations on vectors of length 4 if h^2 is calculated ahead of time, or 16 floating point operations per step. The Adams-Bashforth 3rd order method, when used to extrapolate $\underline{z} = \dot{\underline{x}}_1$, results in 7 arithmetic operations on vectors of length 4, or 28 floating point operations per step. The Adams-Bashforth 3rd order method, when used to extrapolate x_2 results in 7 floating point operations per step. Each calculation of $\ddot{\underline{x}}_1$ and $\dot{x}_2$ results in 31 floating point operations. The Godunov method, then, can be seen to take 82 floating point operations per step, which can be contrasted with the Adams-Bashforth 3rd order method when used to solve equation (2.54) which takes a total of 76 floating point operations per step, or 6 less than the Godunov method.

4th Example: The sine-Gordon Equation

$$\frac{\partial^2\phi}{\partial t^2} = \frac{\partial^2\phi}{\partial x^2} - sin\phi \tag{2.58}$$

A fourth example was attempted to see if the Godunov method would work satisfactorily on a more difficult, nonlinear system. The sine-Gordon equation has been used to describe, among other things, propagation of a crystal dislocation, Bloch wall motion of magnetic crystals and a unitary theory for elementary particles [12]. Solitary wave solutions of the sine-Gordon equation are called solitons, which correspond to a positive sense of rotation, and antisolitons, which correspond to a negative sense of rotation. Solitary wave solutions of the sine-Gordon equation exhibit many of the properties of elementary particles of physics such as conservation of total rotation (they are created and destroyed in pairs) and an invariance to a Lorentz transformation [12]. One other important property of the solitary wave solution is that a soliton and an antisoliton can pass through each other without mutual destruction even though such destruction would not violate conservation of total rotation. Figure (2.4) shows one possible physical representation of the sine-Gordon equation, a rod of infinite length, with pendulums of zero width all immediately next to each other, an1 with a string connecting the bottoms of all the pendulums together. The pendulum action results in the $sin\phi$ term and the string results in the wave action, or the $\frac{\partial^2\phi}{\partial x^2}$ term.

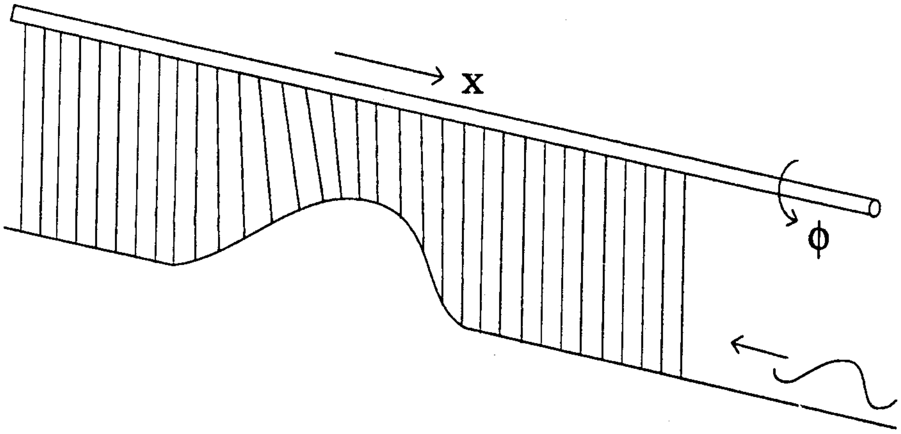


Figure 2.4 A Physical Representation of the sine-Gordon Equation

The sine-Gordon equation can be separated into a set of first or second order ordinary differential equations in exactly the same manner as was the wave equation. Letting $\frac{\partial\phi}{\partial t} = V$, and $\frac{\partial V}{\partial t} = \frac{\partial^2\phi}{\partial x^2} - sin\phi$ and using a centered difference formula to approximate $\frac{\partial^2\phi}{\partial x^2}$, and using boundary conditions of $\phi_0 = 0$ and $\phi_N = 2\pi$, we get the following set of equations:

$$\frac{d\phi_1}{dt} = V_1$$
$$\frac{dV_1}{dt} = \frac{d^2\phi_1}{dt^2} = \frac{1}{(\Delta x)^2}[\phi_2 - 2\phi_1 + \phi_0] - sin\phi_1$$
$$\frac{d\phi_2}{dt} = V_2$$
$$\frac{dV_2}{dt} = \frac{d^2\phi_2}{dt^2} = \frac{1}{(\Delta x)^2}[\phi_3 - 2\phi_2 + \phi_1] - sin\phi_2$$
$$\vdots$$
$$\frac{d\phi_{N-2}}{dt} = V_{N-2}$$
$$\frac{dV_{N-2}}{dt} = \frac{d^2\phi_{N-2}}{dt^2} = \frac{1}{(\Delta x)^2}[\phi_{N-1} - 2\phi_{N-2} + \phi_{N-3}] - sin\phi_{N-2}$$
$$\frac{d\phi_{N-1}}{dt} = V_{N-1}$$
$$\frac{dV_{N-1}}{dt} = \frac{d^2\phi_{N-1}}{dt^2} = \frac{1}{(\Delta x)^2}[\phi_N - 2\phi_{N-1} + \phi_{N-2}] - sin\phi_{N-1} \quad .$$

where N is the number of discrete x values . This set of ODEs can be solved as the first order system $\dot{\underline{x}}(t) = f(\underline{x}(t), t)$ with

$$\underline{x}(t) = \begin{pmatrix} \phi_1(t) \\ \phi_2(t) \\ \vdots \\ \phi_{N-2}(t) \\ \phi_{N-1}(t) \\ V_1(t) \\ V_2(t) \\ \vdots \\ V_{N-2}(t) \\ V_{N-1}(t) \end{pmatrix} \quad ,$$

or it can be solved as the second order system $\ddot{\underline{x}}(t) = f(\underline{x}(t), t)$, with

$$\underline{x}(t) = \begin{pmatrix} \phi_1(t) \\ \phi_2(t) \\ \vdots \\ \phi_{N-2}(t) \\ \phi_{N-1}(t) \end{pmatrix} \quad .$$

Initial conditions can be generated by using the analytic solution for the solitary wave and its first derivative. The analytic solution is used to generate initial conditions for the numerical solution to ensure that the analytic and numerical solutions are solutions to the same problem . The solitary wave solution of the sine-Gordon equation which corresponds to a rotation in ϕ by 2π (as x goes from $-\infty$ to ∞) is

$$\phi(x,t) = 4tan^{-1}[e^{\pm(\frac{x-ut}{\sqrt{1-u^2}})}] \tag{2.59}$$

in which the + sign corresponds to a positive sense of rotation and the pulse can be considered a soliton, and the - sign corresponds to a negative sense of rotation and the pulse can be considered an antisoliton. The first derivative with respect to time of the solitary wave solution is

$$\frac{\partial\phi(x,t)}{\partial t} = \frac{-4ue^{(\frac{x-ut}{\sqrt{1-u^2}})}}{(1+e^{(\frac{2x-2ut}{\sqrt{1-u^2}})})(\sqrt{1+u^2})} \quad . \tag{2.60}$$

The initial conditions for the sine-Gordon equation example are then easily found by plugging $t = 0$ and the desired x-values into equations (2.59) and (2.60), resulting in the equations

$$\phi(x,0) = 4tan^{-1}[e^{(\frac{x}{\sqrt{1-u^2}})}] \quad ,$$

$$\text{and} \quad \frac{\partial\phi(x,0)}{\partial t} = \frac{-4ue^{(\frac{x}{\sqrt{1-u^2}})}}{(1+e^{(\frac{2x}{\sqrt{1-u^2}})})(\sqrt{1-u^2})} \quad .$$

Boundary conditions of

$$\phi(x_0,t) = 0 \quad ,$$

$$\text{and} \quad \phi(x_N;t) = 2\pi$$

must be used because of the rotation from 0 to 2π as x goes from $-\infty$ to ∞.

Stability Analysis for Godunov Integration

A stability analysis will be attempted on a linearized equivalent of equation (2.58). Discretization in the time axis using a Godunov step and in the x-axis using a centered difference scheme results in the equation

$$\frac{\phi_j(t+\Delta t) - 2\phi_j(t) + \phi_j(t-\Delta t)}{(\Delta t)^2} = \frac{\phi_{j+1}(t) - 2\phi_j(t) + \phi_{j-1}(t)}{(\Delta x)^2} - sin\phi_j(t) \quad .$$

where $\phi_j(t)$ corresponds to ϕ at space step j and time step t. Consider the state with ϕ_j very close to 0 or to 2π , then $sin\phi_j \simeq \phi_j$, and

$$\frac{\phi_j(t+\Delta t) - 2\phi_j(t) + \phi_j(t-\Delta t)}{(\Delta t)^2} \simeq \frac{\phi_{j+1}(t) - 2\phi_j(t) + \phi_{j-1}(t)}{(\Delta x)^2} - \phi_j(t) \quad . \qquad (2.59)$$

Seeking a solution in the form of $\phi_j(t) = e^{i(k_j - \omega t)}$ and substituting this into the left hand side of equation (2.59),

$$\begin{aligned} \phi_j(t+\Delta t) - 2\phi_j(t) + \phi_j(t-\Delta t) &= e^{ik_j}[e^{-i\omega(t+\Delta t)} - 2e^{-i\omega t} + e^{-i\omega(t-\Delta t)}] \\ &= e^{i(k_j-\omega t)}[e^{i\omega\Delta t} + e^{-i\omega\Delta t} - 2] \\ &= 2[cos(\omega\Delta t) - 1]e^{i(k_j-\omega t)} \quad , \\ \text{so} \quad \frac{\phi_j(t+\Delta t) - 2\phi_j(t) + \phi_j(t-\Delta t)}{(\Delta t)^2} &= \frac{2[cos(\omega\Delta t) - 1]e^{i(k_j-\omega t)}}{(\Delta t)^2} \quad . \end{aligned} \tag{2.60}$$

Substituting $\phi_j(t) = e^{i(k_j-\omega t)}$ into the right hand side of equation (2.59),

$$\frac{\phi_{j+1}(t) - 2\phi_j(t) + \phi_{j-1}(t)}{(\Delta x)^2} = \frac{2[cos(k_j) - 1]e^{i(k_j-\omega t)}}{(\Delta x)^2} - e^{i(k_j-\omega t)} \quad . \tag{2.61}$$

Substituting the right hand sides of equations (2.60) and (2.61) back into equation (2.59) and dividing through by $e^{i(k_j-\omega t)}$,

$$\begin{aligned} \frac{2[cos(\omega\Delta t) - 1]}{(\Delta t)^2} &= \frac{2[cos(k_j) - 1]}{(\Delta x)^2} - 1 \quad , \\ \text{and} \quad 2[cos(\omega\Delta t) - 1] &= 2[cos(k_j) - 1]\frac{(\Delta t)^2}{(\Delta x)^2} - (\Delta t)^2 \qquad (2.62) \\ \text{so} \quad \omega\Delta t &= cos^{-1}[(cos(k_j) - 1)\frac{(\Delta t)^2}{(\Delta x)^2} - \frac{(\Delta t)^2}{2} + 1] \\ \text{or} \quad \omega &= \frac{1}{\Delta t}cos^{-1}[1 - \frac{(\Delta t)^2}{2} + \frac{(\Delta t)^2}{(\Delta x)^2}(cos(k_j) - 1)] \quad . \end{aligned}$$

Now $cos(k_j) - 1 \leq 0$, so from equation (2.62), $cos(\omega\Delta t) \leq 1 - \frac{(\Delta t)^2}{2} < 1$. The value $k_j = \pi$ causes the worst case of $cos(k_j) - 1 = -2$. For this value of k_j,

$$cos(\omega\Delta t) = 1 - \frac{(\Delta t)^2}{2} - 2\frac{(\Delta t)^2}{(\Delta x)^2} = 1 - 2(\Delta t)^2[\frac{1}{(\Delta x)^2} + \frac{1}{4}] \quad .$$

If $\Delta t = \Delta x$ then $cos(\omega\Delta t) = -1 - \frac{(\Delta t)^2}{2}$ causing a weak instability. If $(\Delta t^2) \leq \frac{1}{\frac{1}{(\Delta x)^2} + \frac{1}{4}} = \frac{(\Delta x)^2}{1 + \frac{(\Delta x)^2}{4}}$ then the Godunov solution should be stable. A truncation error analysis for the sine-Gordon equation is not attempted due to the complexity of the analytic solution.

Floating Point Operations Count

When the sine-Gordon equation is solved using an x-axis extending from -10 to 10 in increments of .1, each second derivative evaluation (needed for a Godunov solution) takes 1,194 floating point operations per step and each first derivative evaluation (needed for an Adams-Bashforth 3rd order solution) takes 1,393 floating point operations per step if the sine function is counted as one floating point operation and if $\frac{1}{(\Delta x)^2}$ is calculated ahead of time. The Godunov method, $\underline{x}_{n+1} = 2\underline{x}_n - \underline{x}_{n-1} + h^2\ddot{\underline{x}}_n$ then results in 4 floating point operations on vectors of length 199, giving a total of 4(199) + 1,194 = 1,990 floating point operations per step for a Godunov solution of the sine-Gordon equation. The Adams-Bashforth 3rd order solution results in 7 floating point operations on vectors of length 398 giving a total of 7(398) + 1,393 = 4,179 floating point operations per step. The Godunov method then has approximately a two to one floating point operations count advantage over the Adams-Bashforth 3rd order solution. This is about the same as what happened when these two methods were used to solve the wave equation and is expected since both the wave and sine-Gordon equations result in first and second order ODEs of the form $\dot{\underline{x}}(t) = \underline{f}(\underline{x}(t), t)$ and $\ddot{\underline{x}}(t) = \underline{f}(\underline{x}(t), t)$, respectively .

CHAPTER 3 - EXPERIMENTAL RESULTS

This chapter will show what happens when the schemes and systems discussed analytically in Chapter 2 are actually coded up and used for simulation runs on a computer. The Godunov integration method was first implemented in Control-C [13], a computer aided control system design software package which is a superset of Matlab [11]. Matlab is a software package designed to perform arithmetic operations conveniently on vectors and matrices. Later on, the integration software was implemented in C to speed up execution time. The graphs and the floating point operation counts shown in this chapter come from data generated using the C code. The errors shown are identical to that resulting from the Control-C code, but the floating point operations counts may be somewhat different. The C code is optimized to not carry out any more floating point operations than necessary. For instance, when the wave equation is solved using the Godunov method, it is not necessary to solve for $\dot{\underline{x}}_1$ or for $\underline{x}_2$. Even though $\dot{\underline{x}}_1$ and $\underline{x}_2$ will be all zeroes in this case, floating point operations will still be accrued, so the C code simply doesn't solve for these two variables when solving the wave equation. The Control-C code does not make a provision for this contingency. The Control-C code and the C (VAX-C) code are shown in their entirety in Appendix B. First, a fragment of the Control-C code used to solve the systems is shown.

As stated in Chapter 2, the most general form of the problem to be solved using the Godunov integration technique is

$$\ddot{\underline{x}}_1 = \underline{f}_1(\underline{x}_1, \dot{\underline{x}}_1, \underline{x}_2, \underline{u}, t) \tag{3.1}$$

$$\dot{\underline{x}}_2 = \underline{f}_2(\underline{x}_1, \dot{\underline{x}}_1, \underline{x}_2, \underline{u}, t) \tag{3.2}$$

$$\underline{y} = \underline{g}(\underline{x}_1, \dot{\underline{x}}_1, \underline{x}_2, \underline{u}, t) \quad . \tag{3.3}$$

It can be seen that an extra calculation will be required to solve for $\dot{\underline{x}}_1$, since $\dot{\underline{x}}_1$ is required explicitly in the right hand side of the system equations, and yet there is not a system equation to use to evaluate $\dot{\underline{x}}_1$. This results in the addition of a third equation to the general problem:

$$\frac{d}{dt}\dot{\underline{x}}_1 = \ddot{\underline{x}}_1 = f_1(\underline{x}_1,\dot{\underline{x}}_1,\underline{x}_2,\underline{u},t) \quad . \tag{3.4}$$

If equation (3.1) is solved with a Godunov step, and equations (3.2) and (3.4) are solved using Adams-Bashforth 3rd order steps, then three simultaneous equations result:

$$\begin{aligned} \underline{x}_{1_{n+1}} &= 2\underline{x}_{1_n} - \underline{x}_{1_{n-1}} + h^2\ddot{\underline{x}}_{1_n} \\ \dot{\underline{x}}_{1_{n+1}} &= \dot{\underline{x}}_{1_n} + \frac{h}{12}[23\ddot{\underline{x}}_{1_n} - 16\ddot{\underline{x}}_{1_{n-1}} + 5\ddot{\underline{x}}_{1_{n-2}}] \\ \underline{x}_{2_{n+1}} &= \underline{x}_{2_n} + \frac{h}{12}[23\dot{\underline{x}}_{2_n} - 16\dot{\underline{x}}_{2_{n-1}} + 5\dot{\underline{x}}_{2_{n-2}}] \quad . \end{aligned} \tag{3.5}$$

Once the startup is completed, the main loop of the computer code will need to calculate the derivatives defined by equations (3.1) and (3.2) and then extrapolate forward in time to the next step using equations (3.5).

In Control-C, the code appears as follows:

```
for   i = 3 : n,...

    [x1dd] = f1(a,b,c,d,x1,x1d,x2,u,i);...   //   $\ddot{\underline{x}}_{1_i}$

    [x2d] = f2(e,f,g,h,x1,x1d,x2,u,i);...   //   $\dot{\underline{x}}_{2_i}$

    [yt] = g(ii,j,k,l,x1,x1d,x2,u,i);...   //   output of system at step i

    y(:,i) = yt;...   //   this adds the output vector $\underline{yt}_i$ to the output matrix y

    x1vo = x1o;...   //   $\underline{x}_{1_{i-1}}$

    x1o = x1;...   //   $\underline{x}_{1_i}$
```

```
        [x1] = GODSTEP(x1o, x1vo, x1dd, dt); ...   //   $\underline{x}_{1_{i+1}}$
        [x1d] = AB3(x1d, x1dd, x1ddo, x1ddvo, dt); ...   //   $\dot{\underline{x}}_{1_{i+1}}$
        [x2] = AB3(x2, x2d, x2do, x2dvo, dt); ...   //   $\underline{x}_{2_{i+1}}$
        x1ddvo = x1ddo; ...   //   $\ddot{\underline{x}}_{1_{i-1}}$
        x1ddo = x1dd; ...   //   $\ddot{\underline{x}}_{1_i}$
        x2dvo = x2do; ...   //   $\dot{\underline{x}}_{2_{i-1}}$
        x2do = x2d; ...   //   $\dot{\underline{x}}_{2_i}$
    end
```

where *d* and *dd* stand for dot and double dot, and *o* and *vo* stand for old and very old respectively. The sine-Gordon equation was not solved with Control-C since the Control-C code was written to accept linear systems. The derivative evaluations used the Control-C functions

```
        //[xdd] = f1(a, b, c, d, x1, x1d, x2, u, t)
        xdd = a * x1d + b * x1 + c * x2 + d * u(t);

and     //[x2d] = f2(e, f, g, h, x1, x1d, x2, u, t)
        x2d = e * x1d + f * x1 + g * x2 + h * u(t);    ,
```

where a through h are defined by the system to be solved. Although f_1 and f_2 are identical functions except in name, they were kept separate for clarity of the software. The forward extrapolations in time for $\underline{x}_1$, $\dot{\underline{x}}_1$ and $\underline{x}_2$ use the Control-C functions

$$//[xn] = GODSTEP(x, xo, xdd, dt)$$

$$xn = 2 * x - xo + (dt * dt * xdd);$$

and $$//[xn] = AB3(x, xd, xdo, xdvo, dt)$$

$$xn = x + (dt/12) * (23 * xd - 16 * xdo + 5 * xdvo); \quad .$$

The remainder of Chapter 3 will show:

- Analytic solutions
- Absolute errors
- Number of floating point operations required

for the wave equation, the human body model, the passive electrical system and the sine-Gordon equation. The derivations of the analytic solutions are shown in Appendix C. The C code, which was used to generate the absolute error graphs and the floating point operations counts, also had a Runge-Kutta 4th order first derivative system solver, so the errors resulting from the Runge-Kutta 4th order solution are included on some of the graphs.

FIRST EXAMPLE - WAVE EQUATION

The analytic solution for the hyperbolic partial differential equation

$$\frac{\partial^2 U}{\partial t^2} = \frac{\partial^2 U}{\partial x^2}$$

with initial conditions

$$\frac{\partial U(x,0)}{\partial t} = 0 \quad \text{and} \quad U(x,0) = sin(\frac{\pi}{2})$$

and boundary conditions

$$U(0,t) = 0 \quad , \quad \frac{\partial U(1,t)}{\partial x} = 0$$

is $U(x,t) = sin(\frac{\pi}{2}x)cos(\frac{\pi}{2}t)$. This is shown graphically at the value x=1 in Figure (3.1). The Godunov method will be used to solve the system $\ddot{\underline{x}}(t) = -\omega^2 \underline{x}(t)$ for the wave equation where

$$\omega^2 = \begin{pmatrix} 200 & -100 & 0 & 0 & \dots & 0 \\ -100 & 200 & -100 & 0 & \dots & 0 \\ 0 & -100 & 200 & -100 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & -200 & 200 \end{pmatrix} .$$

The Adams-Bashforth 3rd order and other first order system solvers will solve the system $\dot{\underline{x}}(t) = A\underline{x}(t)$ where

$$A = \begin{pmatrix} 0 & I \\ -\omega^2 & 0 \end{pmatrix} .$$

Figure (3.2) shows the Godunov absolute error using both the Forward Euler and the Taylor Series Expansion startups at the stepsize $h = .01$. Figure (3.3) shows the Adams-Bashforth 3rd order and the Runge-Kutta 4th order errors along with the Godunov errors.

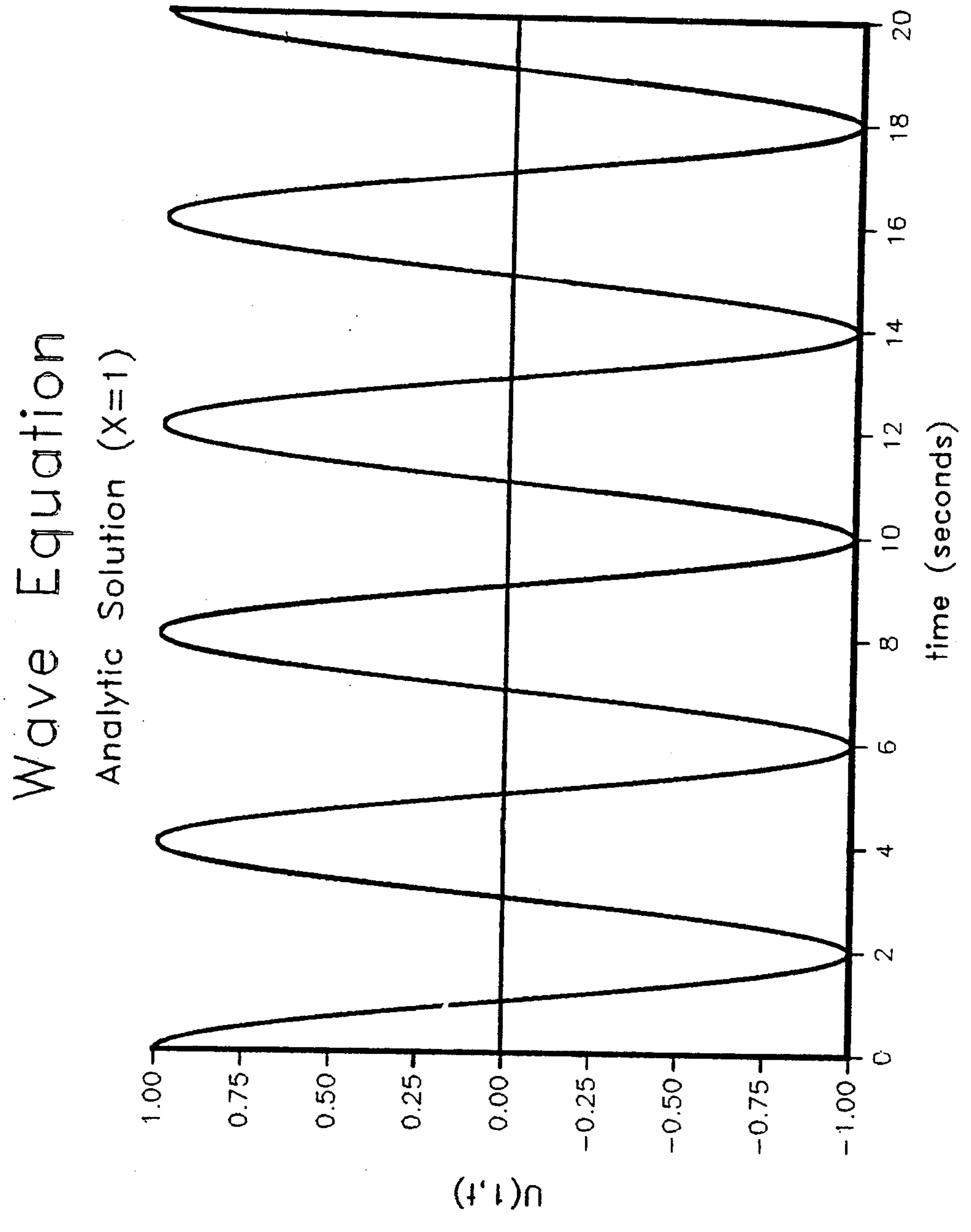


Figure 3.1 Wave Equation (Analytic Solution)

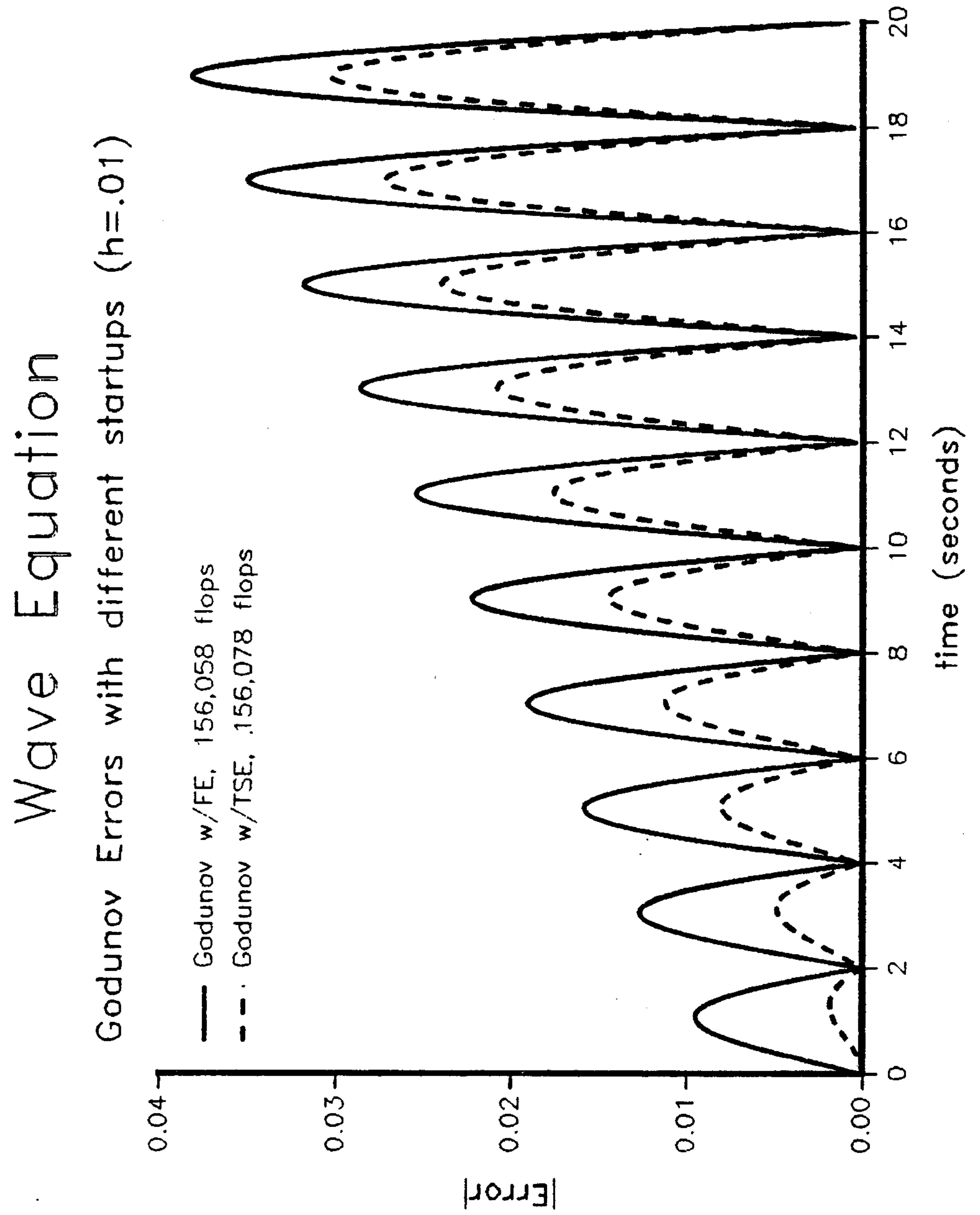


Figure 3.2 Wave Equation ($h = .01$)

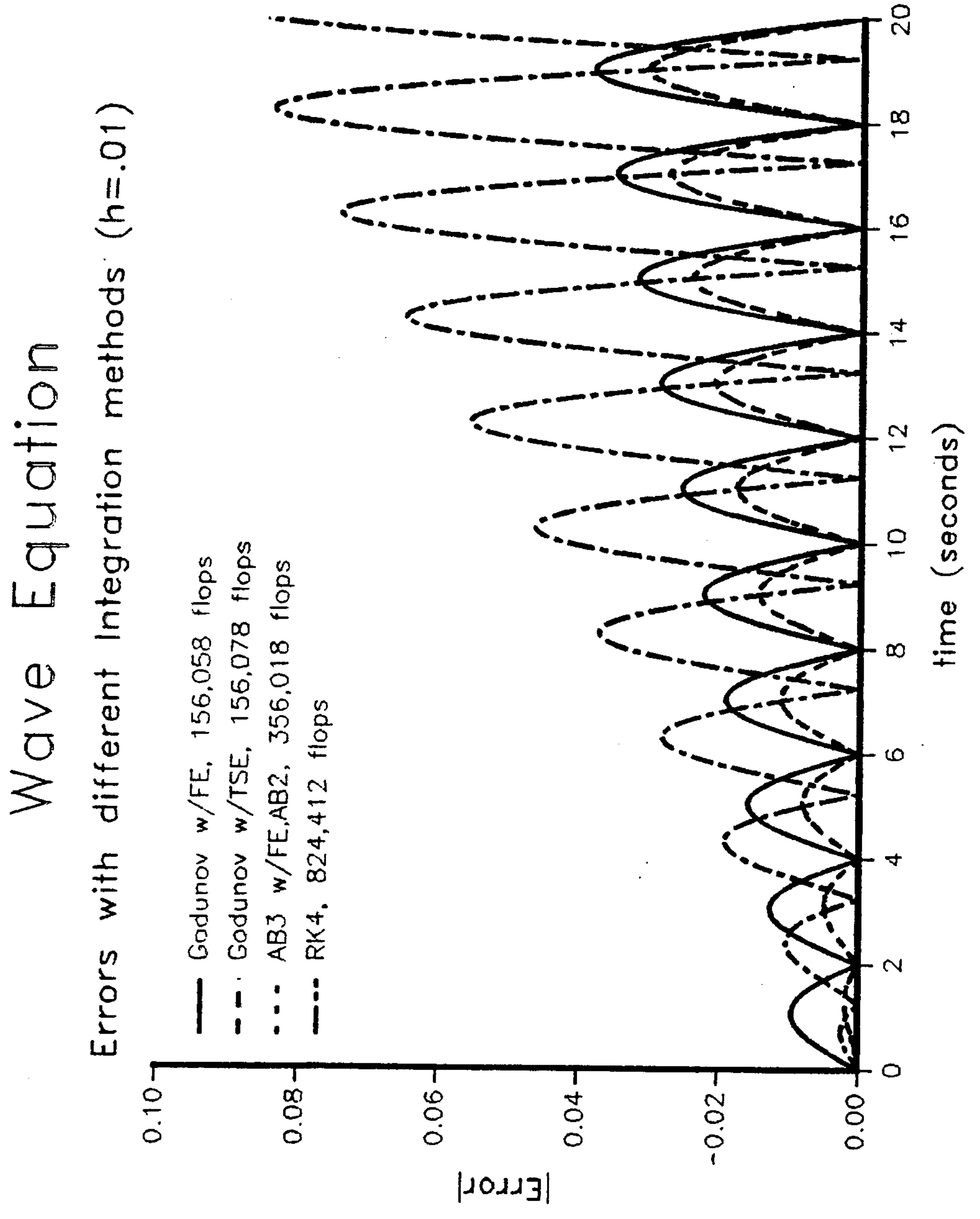


Figure 3.3 Wave Equation ($h = .01$)

Only three traces show on Figure (3.3) because the Godunov method with Taylor Series Expansion startup and the Adams-Bashforth 3rd order method exhibit nearly identical discretization errors. The Godunov method with Forward Euler startup shows a larger error than with the Taylor Series Expansion startup because of the additional term used on the Taylor Series Expansion. One other point to notice on this graph is that the errors seem to be increasing in a linear fashion for all four methods, indicating a weak numerical instability. Floating point operations counts show the expected results with the Adams-Bashforth 3rd order method taking approximately twice as many floating point operations as does the Godunov method.

In an attempt to determine the limits of stability of the Godunov method, the stepsize h was gradually increased. As shown in Figures (3.4) through (3.6) , the magnitude of the error of the Godunov step with Forward Euler startup increases but the rate of increase of this error decreases, while the error with a Taylor Series Expansion startup steadily decreases. The Adams-Bashforth 3rd order method and the Runge-Kutta 4th order methods become unstable at a stepsize less than $h = .05$. When $h = .1$, which is equal to Δx, the Godunov error with a Forward Euler startup remains at a constant value while the error with a Taylor Series Expansion startup nearly vanishes as seen in Figures (3.7) and (3.8) .These errors agree with the results of Chapter 2 which showed that the truncation error due to the Godunov step with a Taylor Series Expansion startup step should vanish when $\Delta t = \Delta x$, and the truncation error of the Godunov method when used with a Forward Euler startup step should be equal to the error of the Forward Euler startup step only. This explains why the error shown in Figure (3.7) remains at a constant value, and indicates that the error shown in Figure (3.8) is purely the result of the machine roundoff error.

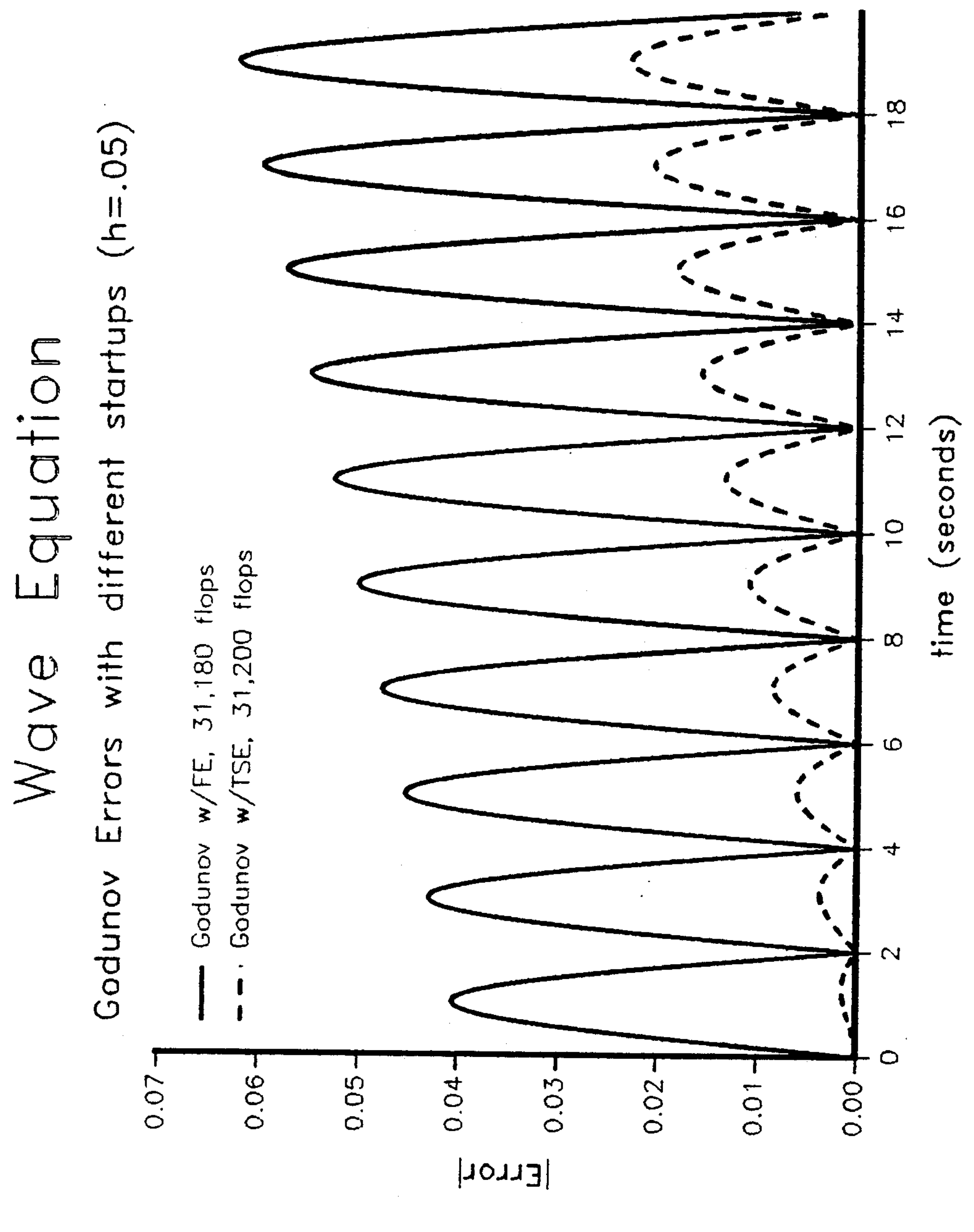


Figure 3.4 Wave Equation (h = .05)

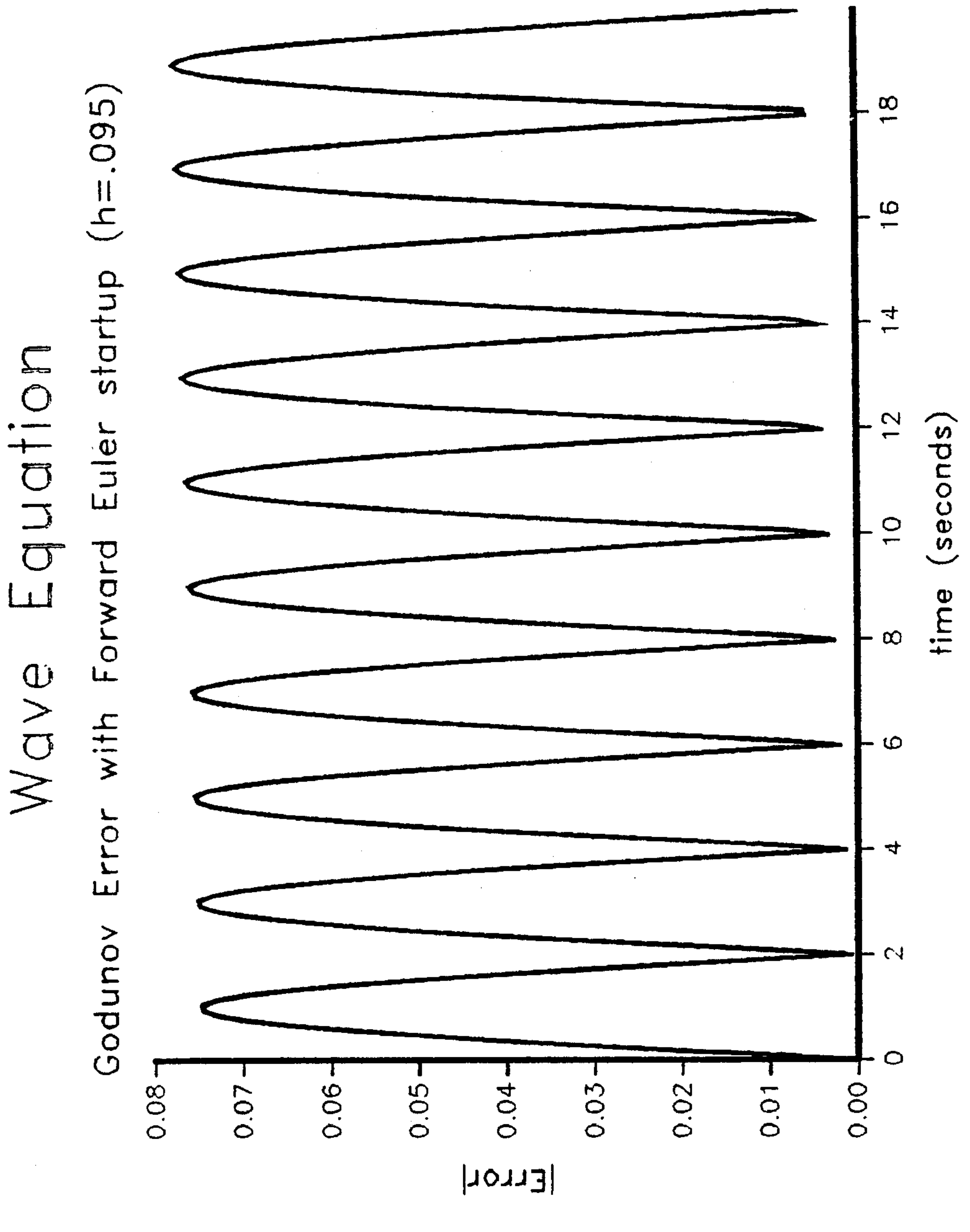


Figure 3.5 Wave Equation ($h = .095$)

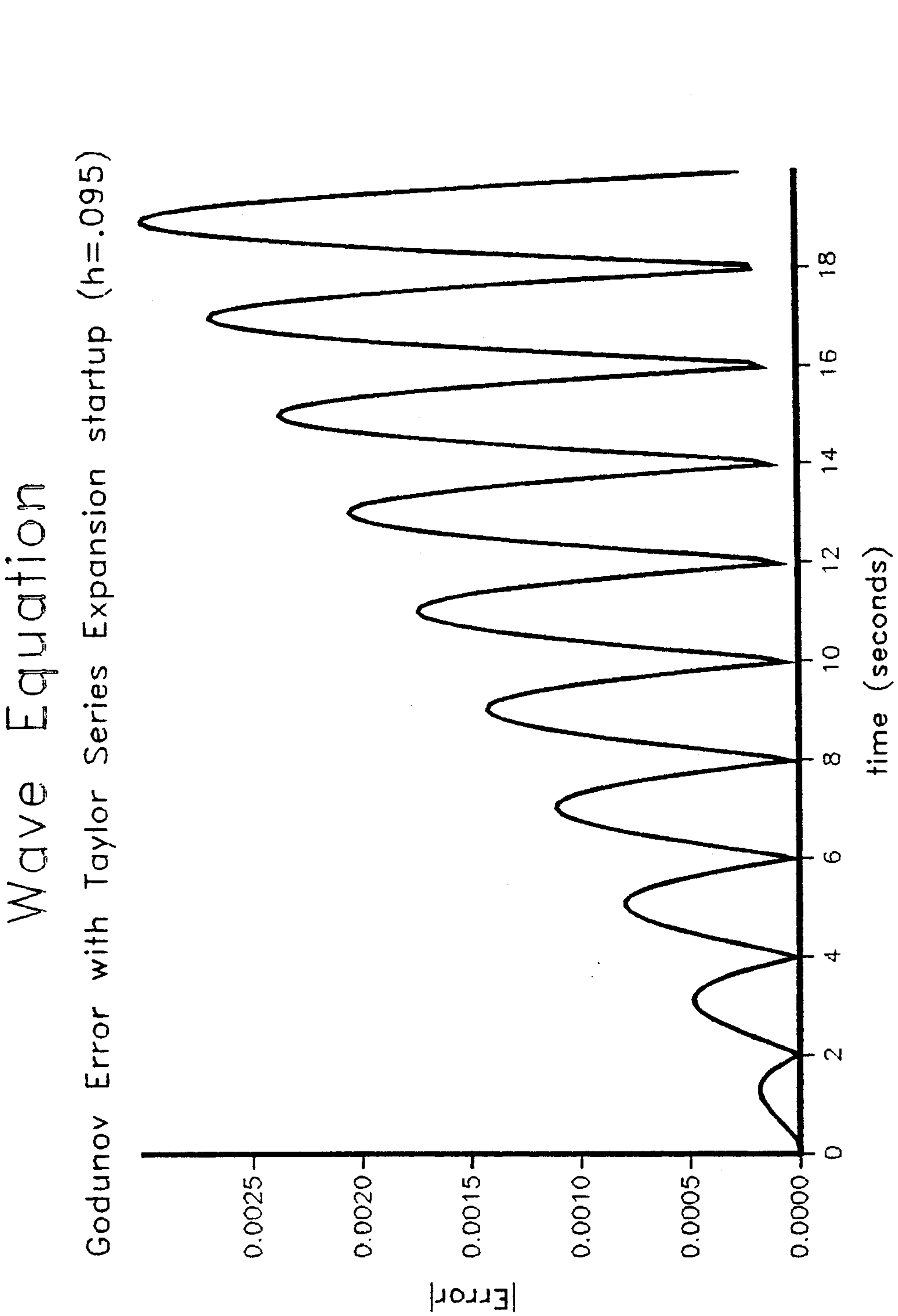


Figure 3.6 Wave Equation ($h = .095$)

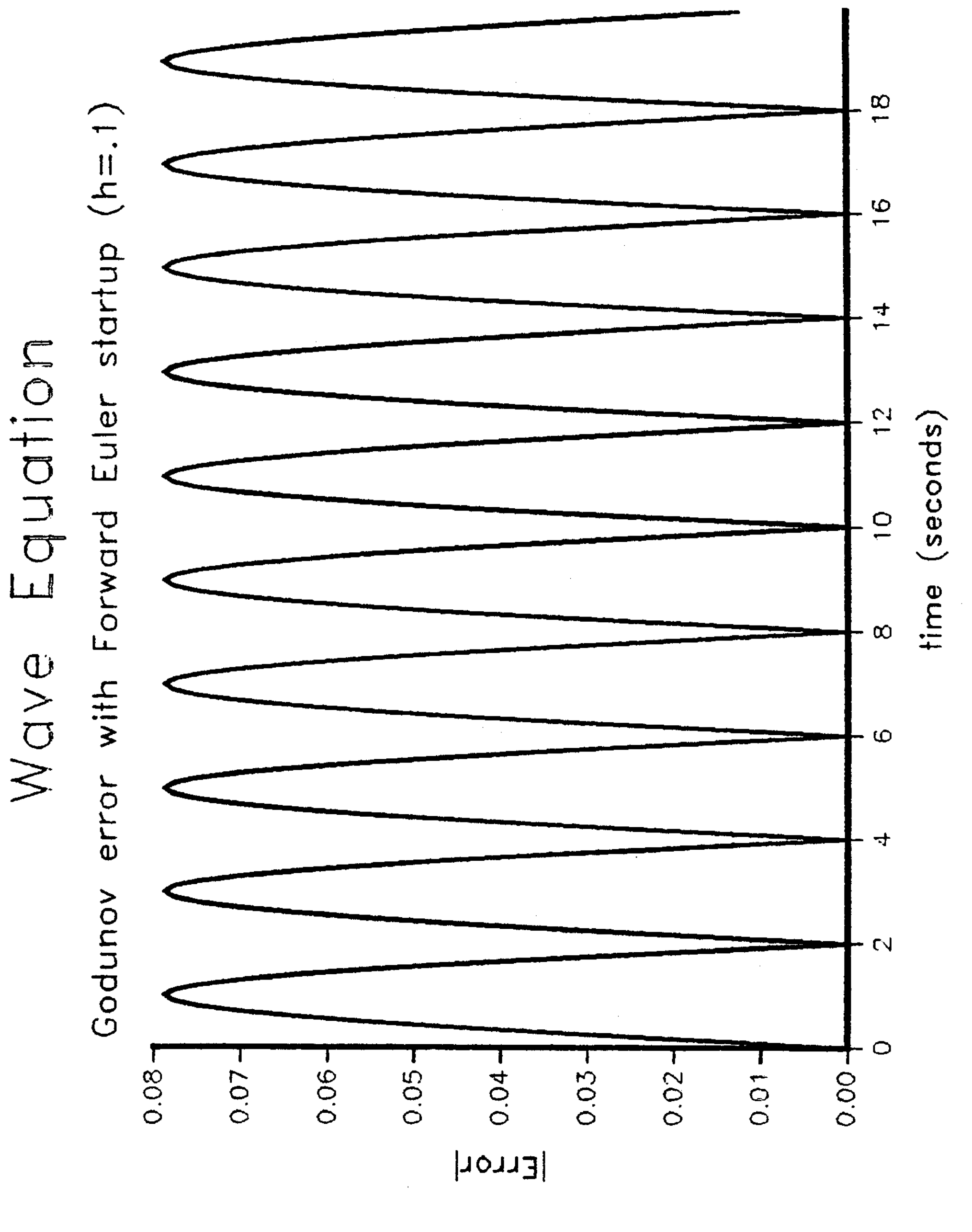


Figure 3.7 Wave Equation ($h = .1$)

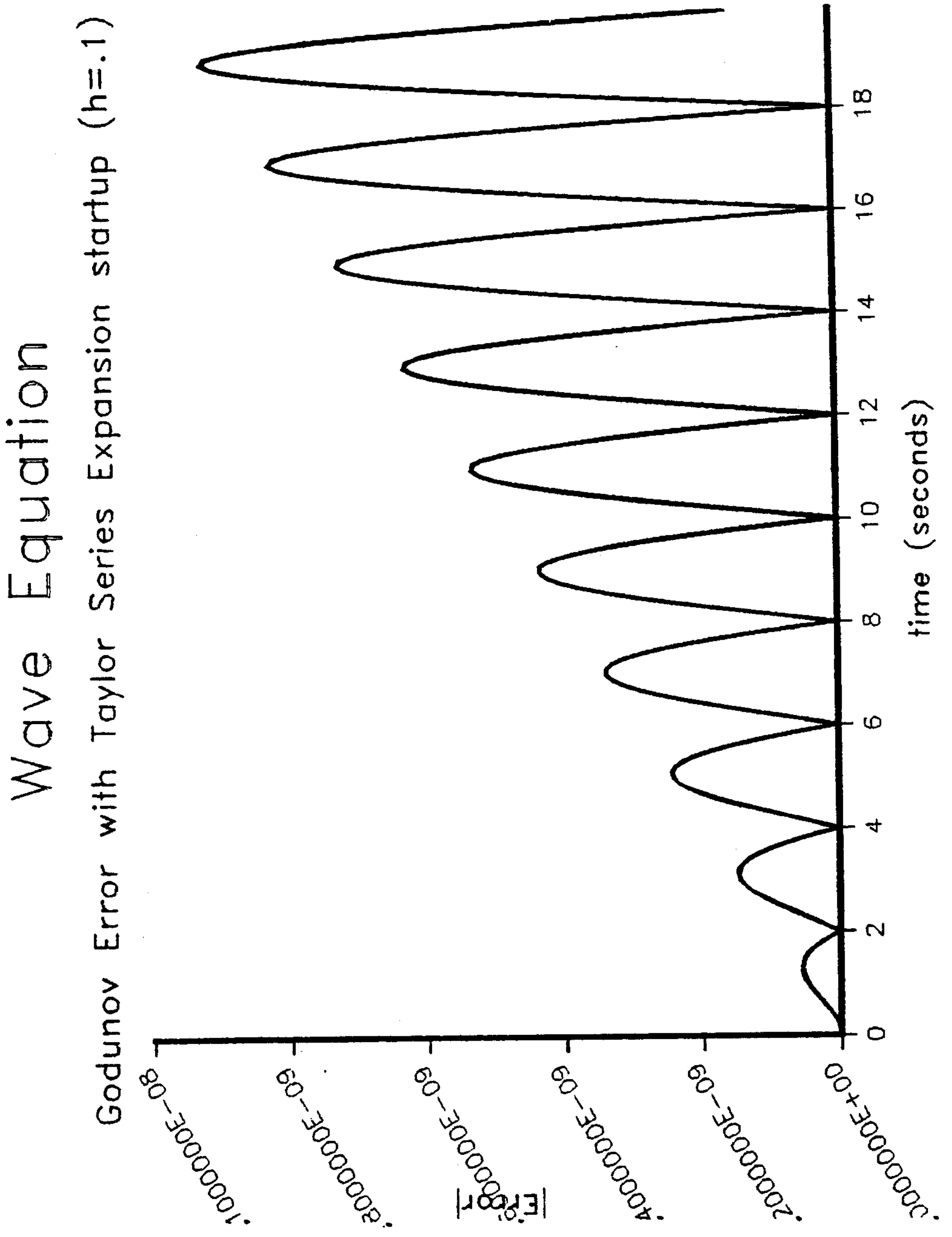


Figure 3.8 Wave Equation ($h = .1$)

Figures (3.9) and (3.10) show that as h is increased slightly above Δx the error resulting from the Godunov step with a Taylor Series Expansion startup starts increasing again due to the increasing error of both the Taylor Series Expansion step and of the Godunov step. The increase in error of the Godunov step with a Forward Euler startup is still dominated by the error of the Forward Euler startup step for this value of h. The Godunov method goes completely unstable when h is increased very much above .1005 as shown in Figure (3.11). This agrees with the stability analysis done in Chapter 2, which shows that for stability the eigenvalues of ω^2 and the stepsize h should satisfy the relationship $h < \frac{2}{\lambda_j}$. Since the largest eigenvalue of ω^2 has a value of $\lambda_j = 19.938347$, h should be less than .1003 to guarantee stability. The discretization error results, the floating point operations counts, and the stability domain results seem to give a clear advantage to the Godunov method over the first order system solvers when used to solve the wave equation.

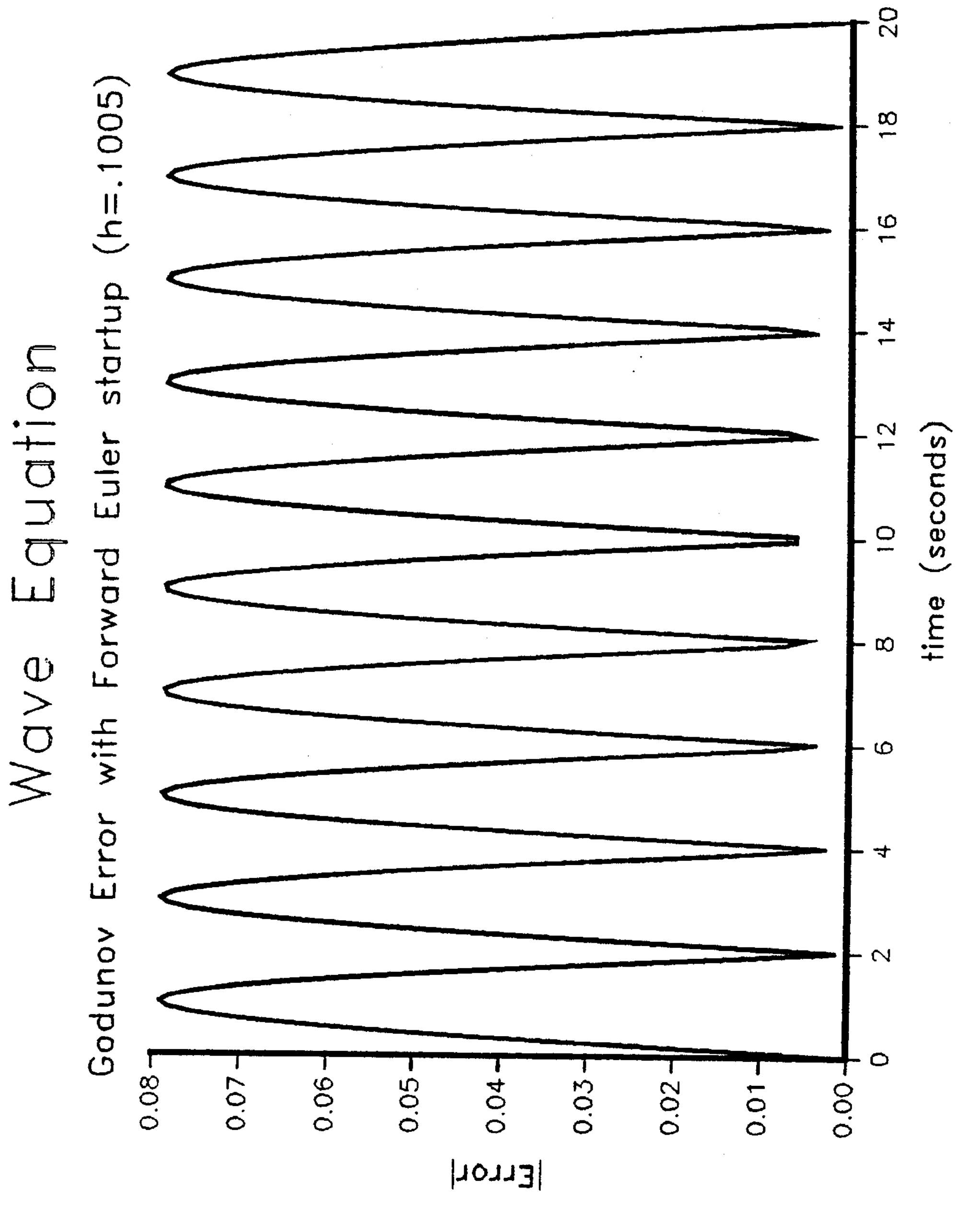


Figure 3.9 Wave Equation ($h = .1005$)

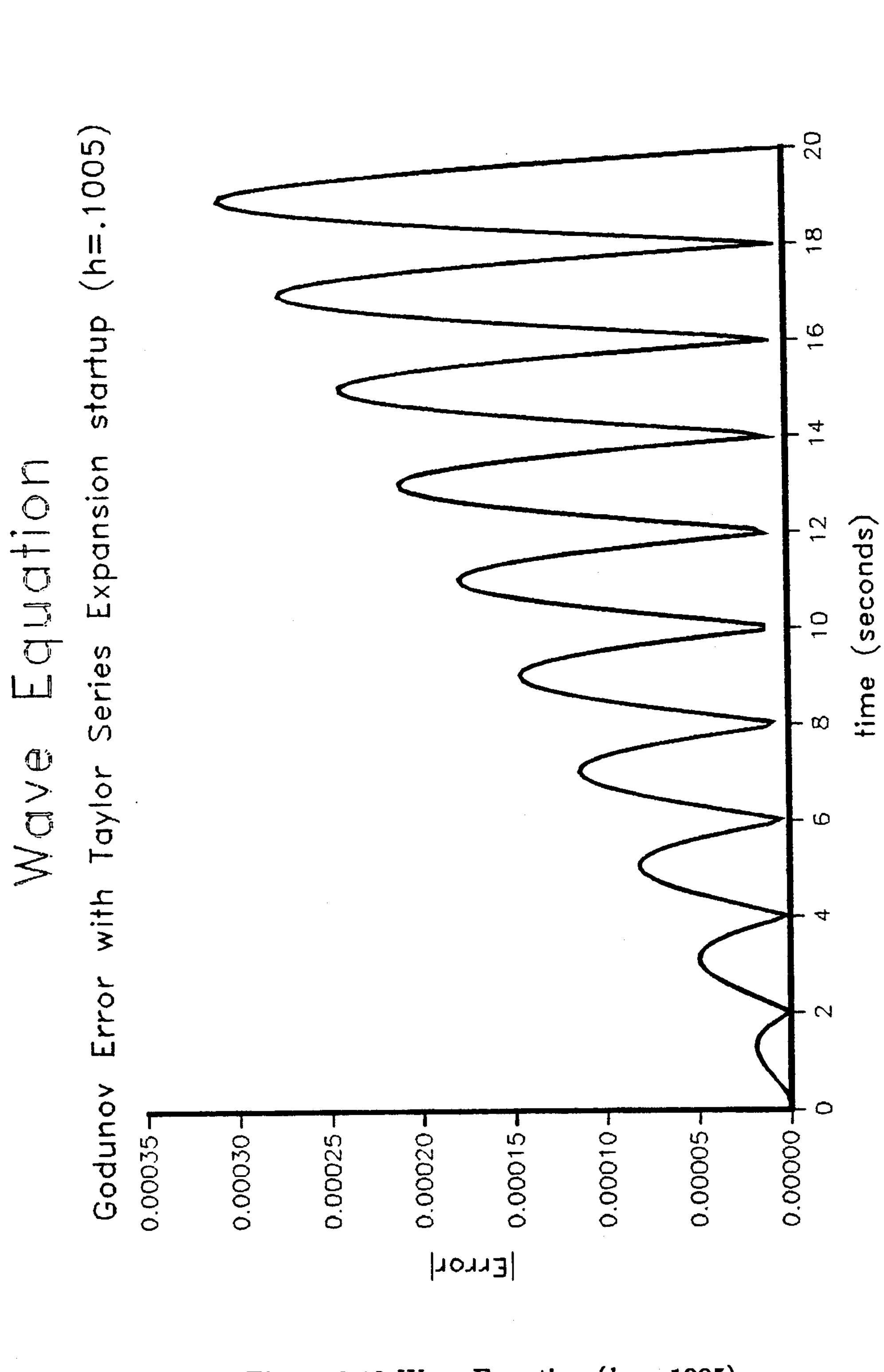


Figure 3.10 Wave Equation ($h = .1005$)

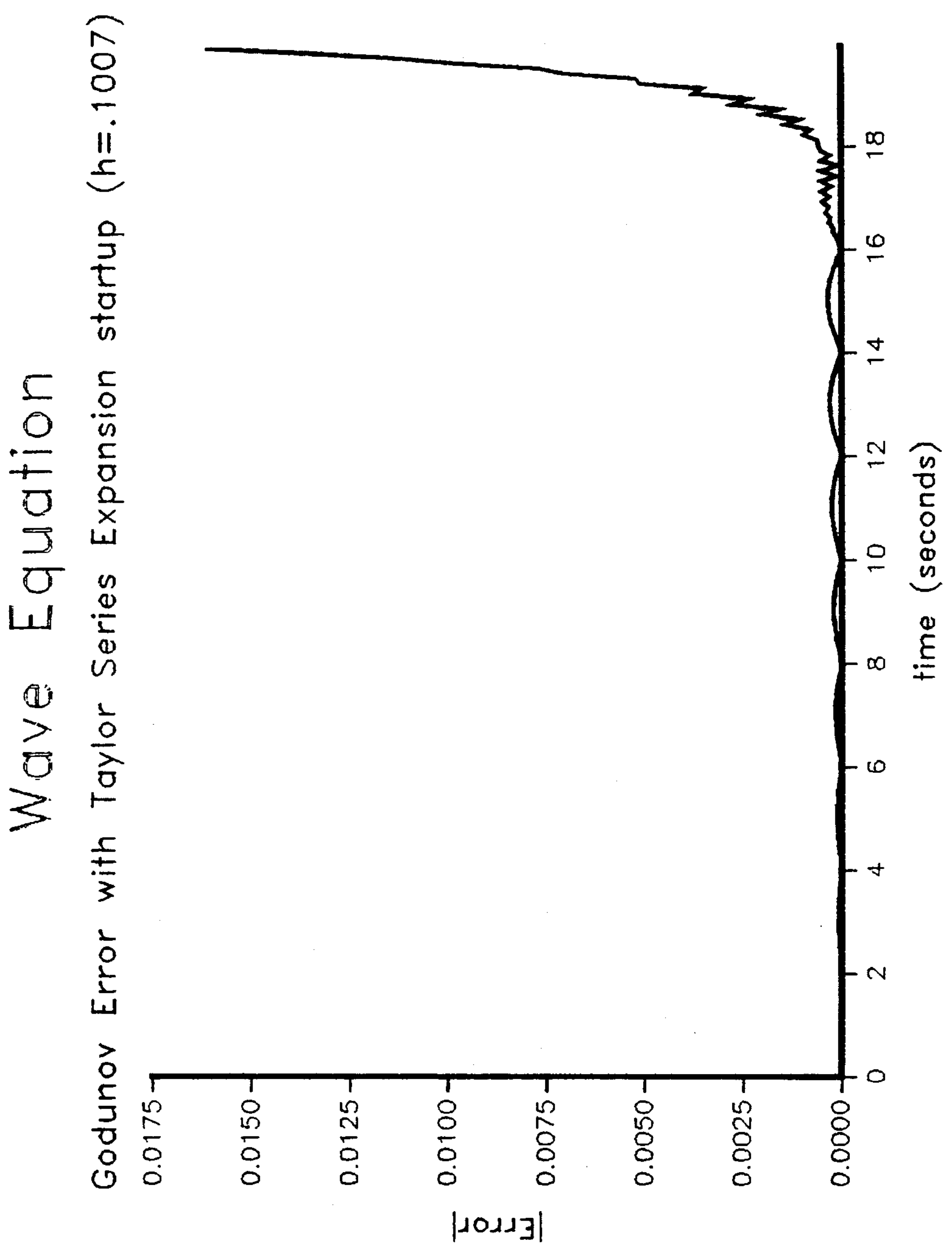


Figure 3.11 Wave Equation (h = .1007)

SECOND EXAMPLE - HUMAN BODY MODEL

The set of second order differential equations to be solved for the human body model is

$$\underline{\ddot{x}}(t) = -\omega^2 \underline{x}(t) - C\underline{\dot{x}}(t) + \underline{d}u(t)$$

where

$$\omega^2 = \begin{pmatrix} \frac{k_1}{m_1} & \frac{-k_1}{m_1} & 0 & 0 \\ \frac{-k_1}{m_2} & \frac{k_1+k_2+k_3}{m_2} & \frac{-k_2}{m_2} & \frac{-k_3}{m_2} \\ 0 & \frac{-k_2}{m_3} & \frac{k_2}{m_3} & 0 \\ 0 & \frac{-k_3}{m_4} & 0 & \frac{k_3}{m_4} \end{pmatrix} ,$$

$$C = \begin{pmatrix} \frac{b_1}{m_1} & \frac{-b_1}{m_1} & 0 & 0 \\ \frac{-b_1}{m_2} & \frac{b_1+b_2+b_3}{m_2} & \frac{-b_2}{m_2} & \frac{-b_3}{m_2} \\ 0 & \frac{-b_2}{m_3} & \frac{b_2}{m_3} & 0 \\ 0 & \frac{-b_3}{m_4} & 0 & \frac{b_3}{m_4} \end{pmatrix}$$

$$\text{and} \quad \underline{d} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{m_4} \end{pmatrix} .$$

First order system solvers were given the equation $\underline{\dot{x}}(t) = A\underline{x}(t) + \underline{b}u(t)$ to solve where

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{k_1}{m_1} & -\frac{b_1}{m_1} & \frac{k_1}{m_1} & \frac{b_1}{m_1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{k_1}{m_2} & \frac{b_1}{m_2} & \frac{-k_1-k_2-k_3}{m_2} & \frac{-b_1-b_2-b_3}{m_2} & \frac{k_2}{m_2} & \frac{b_2}{m_2} & \frac{k_3}{m_2} & \frac{b_3}{m_2} \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{k_2}{m_3} & \frac{b_2}{m_3} & -\frac{k_2}{m_3} & -\frac{b_2}{m_3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{k_3}{m_4} & \frac{b_3}{m_4} & 0 & 0 & -\frac{k_3}{m_4} & -\frac{b_3}{m_4} \end{pmatrix}$$

$$\text{and} \quad \underline{b} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{m_4} \end{pmatrix}$$

The analytic solution to this system is

$$\begin{aligned} y(t) = x_1(t) - x_2(t) = {} & .1306e^{-.3543t} - .0038e^{-3.095t} \\ & + 2e^{-.3326t}(.0524cos(.3638t) - .19067sin(.3638t)) \\ & - 2e^{-.903t}(.112cos(.27t) + .02325sin(.27t)) \\ & + 2(-.00374cos(\pi t) + .0029sin(\pi t)) \end{aligned}$$

when the input $u(t) = 10sin(\pi t)$ is used and is shown in Figure (3.12). The cervical vertebrae are in tension during the negative portions of the solution and are under compression when the solution is greater than zero. Figure (3.13) shows errors for the Godunov, Adams-Bashforth 3rd order and the Runge-Kutta 4th order solutions at $h = .01$. The Runge-Kutta 4th order error is so much bigger than the other two at this point that another graph (Figure 3.14) was made to show just the Godunov and Adams-Bashforth 3rd order errors. Figure (3.14) shows the Godunov method to have a smaller error initially but as t increases the Adams-Bashforth 3rd order error becomes the smaller of the two. When h is increased to a value of 0.1 the Runge-Kutta 4th order error is still much larger than the Godunov and Adams-Bashforth 3rd order errors, and at this step size the Godunov error remains smaller than the Adams-Bashforth 3rd order error with increasing time (Figure 3.15). The Godunov method requires approximately 10% fewer flops than the Adams-Bashforth 3rd order method for this system. The Godunov method goes unstable for $h > 0.14$

while the Adams-Bashforth 3rd order method goes unstable at the slightly larger value of $h = 0.18$ (Figures 3.16 and 3.17). Overall, the Godunov method offers somewhat of an advantage over the Adams-Bashforth 3rd order method in terms of floating point operations counts and discretization errors and a considerably larger advantage over the Runge-Kutta 4th order method in both of these quantities. The Adams-Bashforth 3rd order method does, however, offer a slightly larger stability domain for this system.

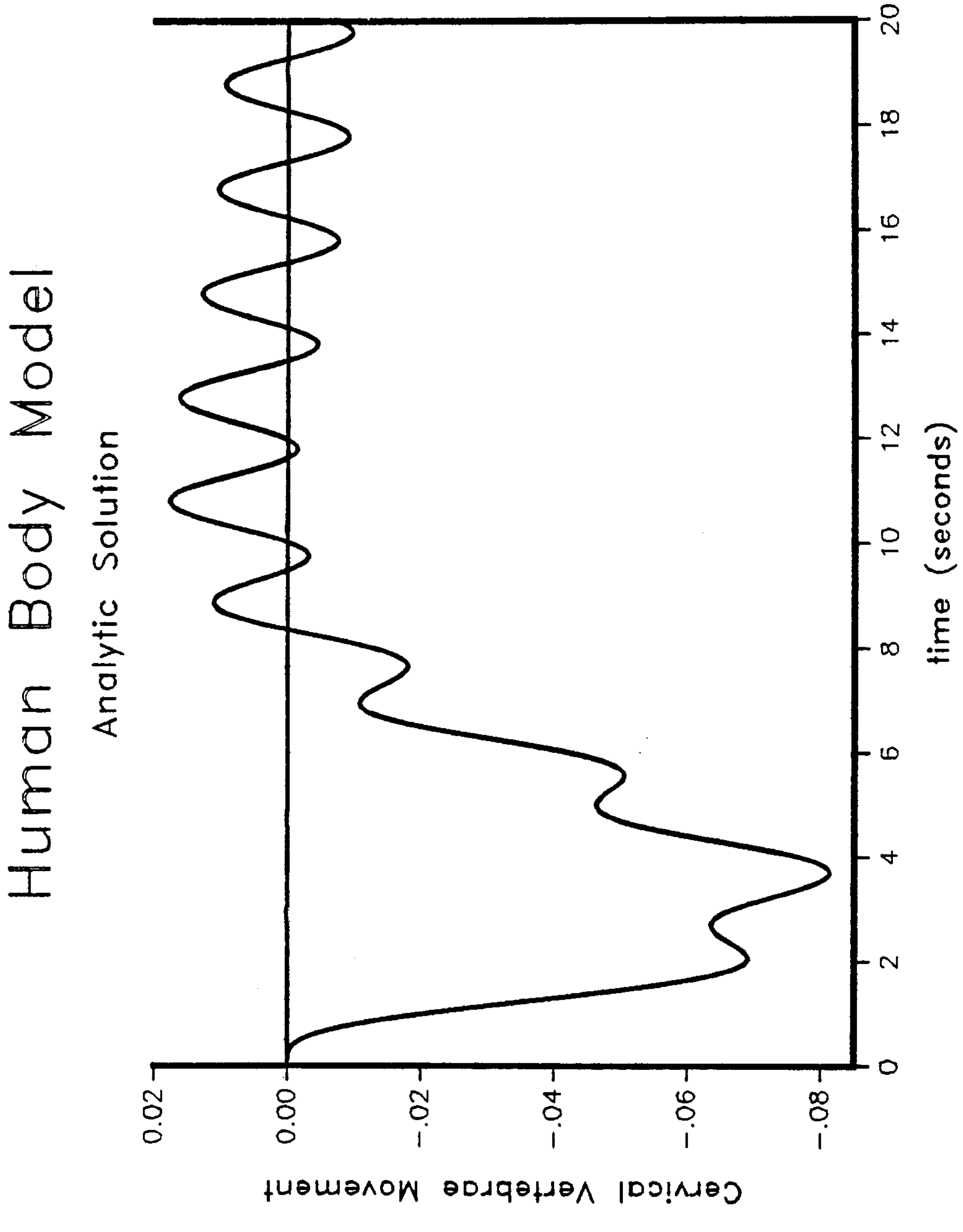


Figure 3.12 Human Body Model (Analytic Solution)

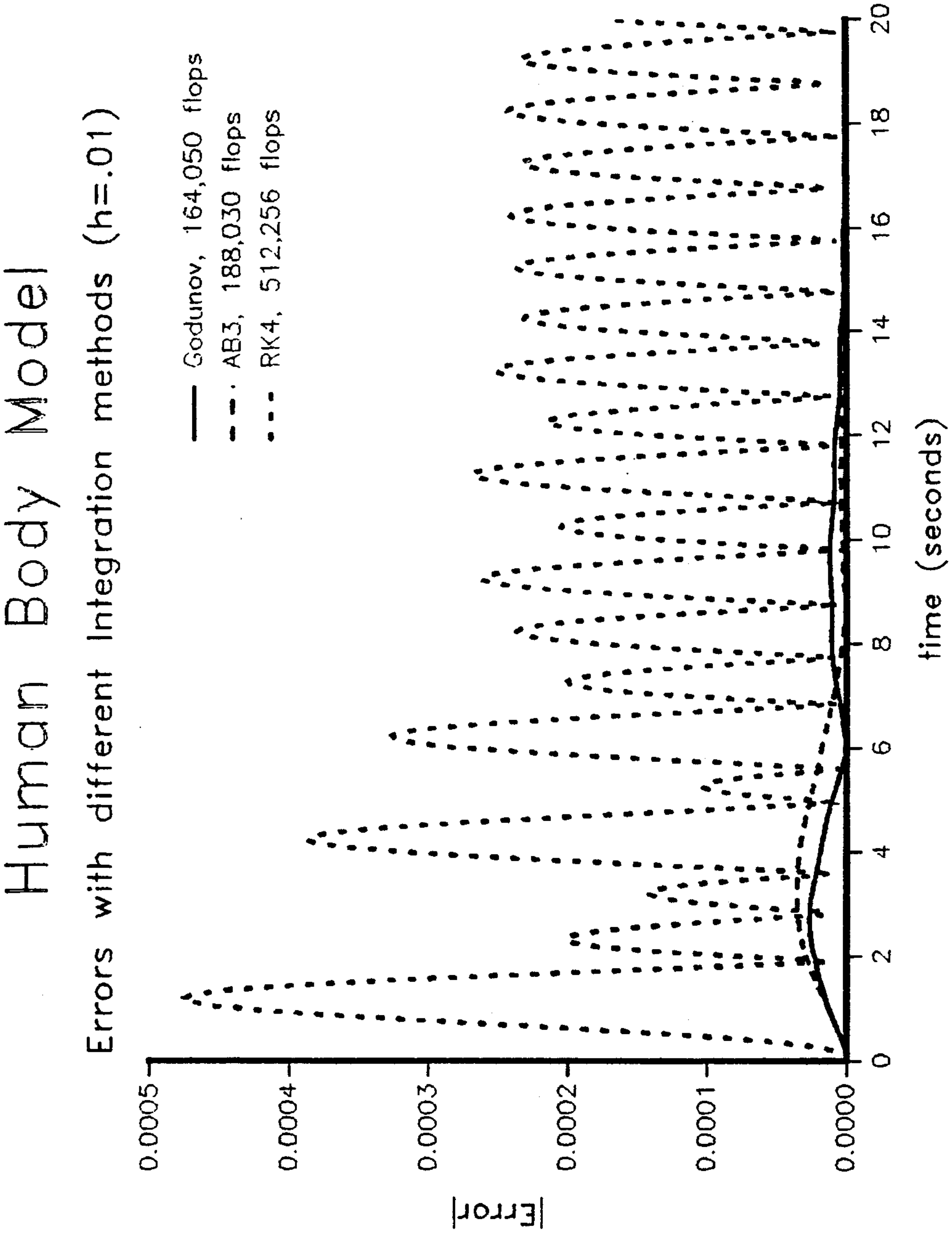


Figure 3.13 Human Body Model ($h = .01$)

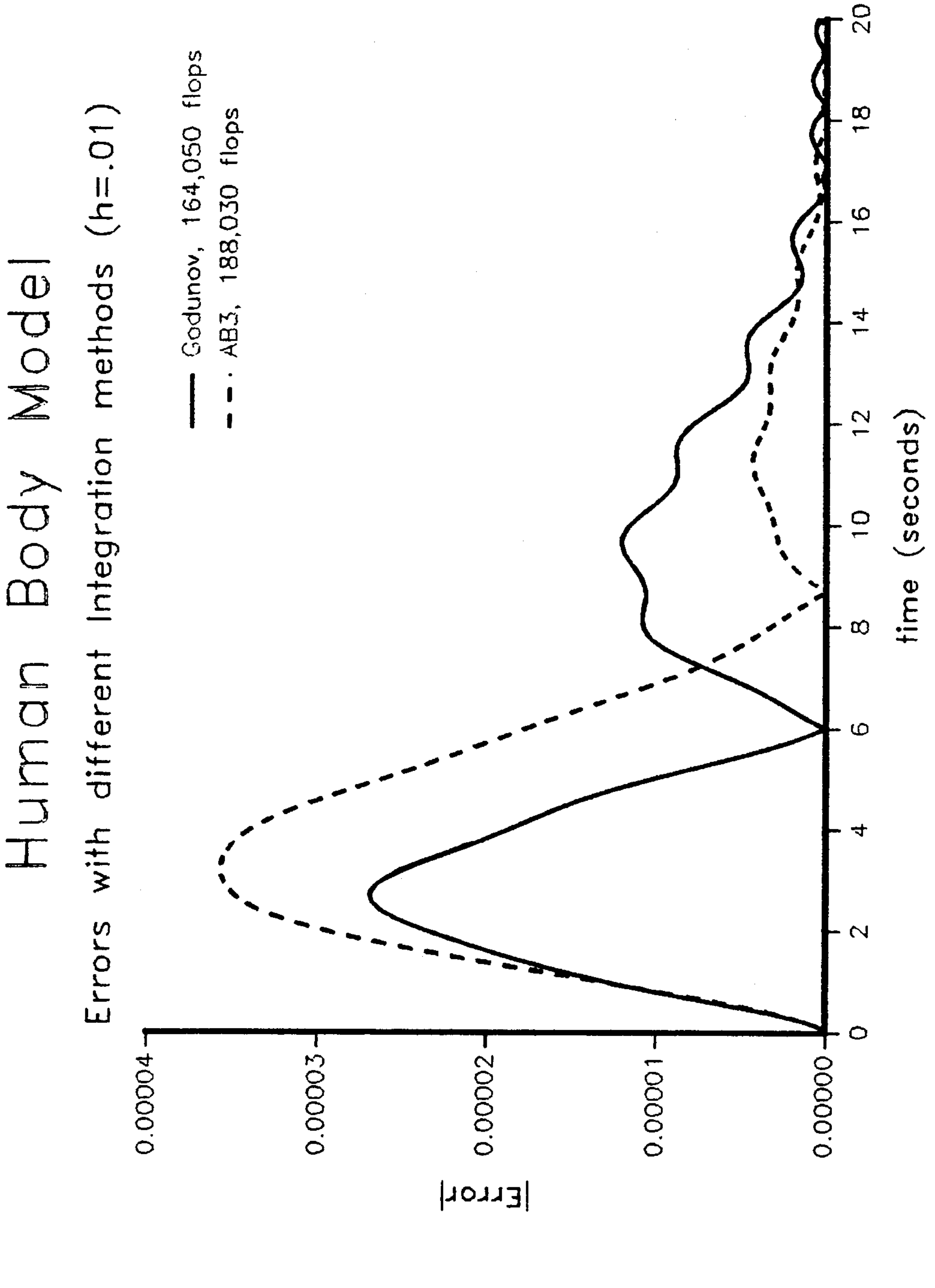


Figure 3.14 Human Body Model ($h = .01$)

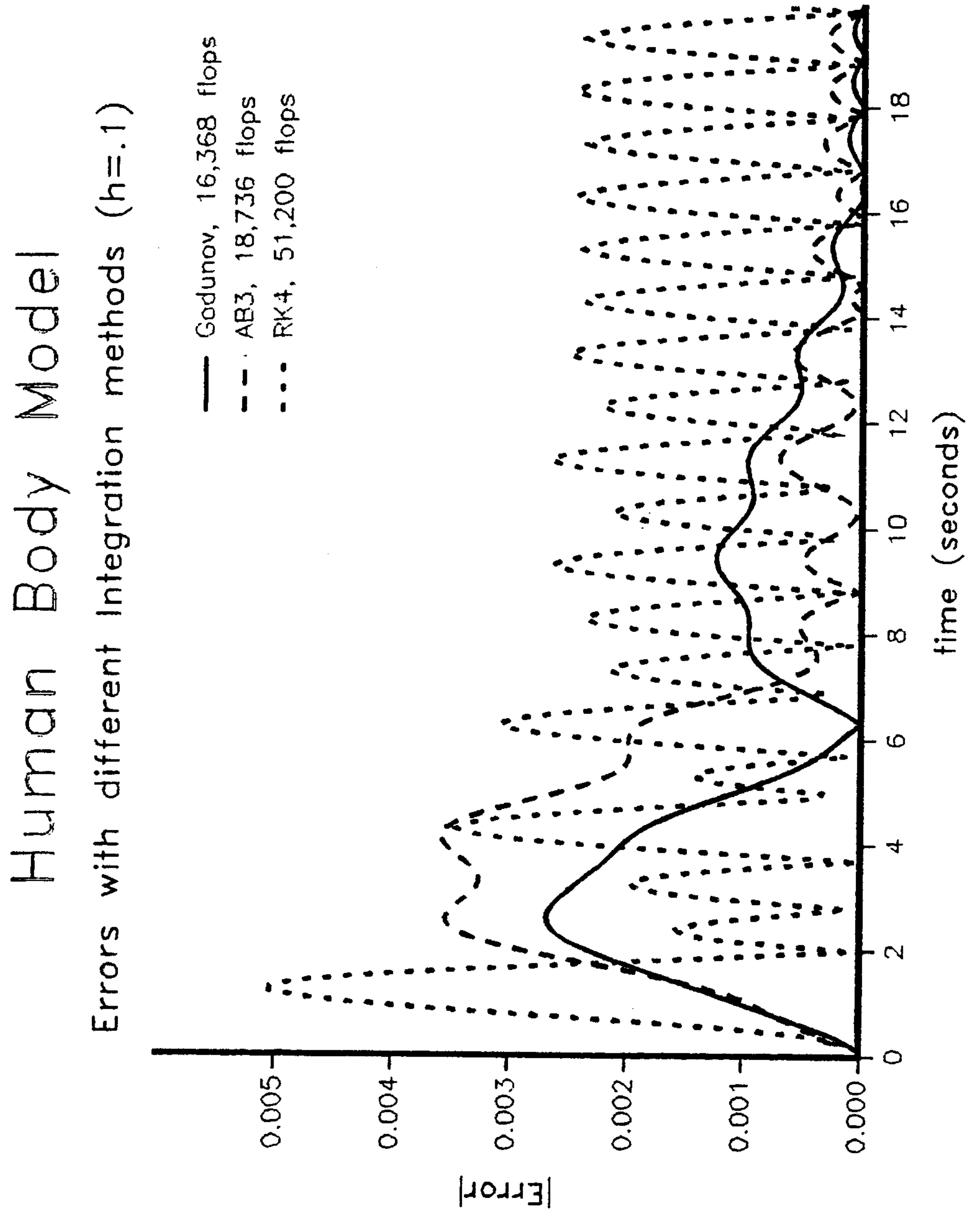


Figure 3.15 Human Body Model ($h = .1$)

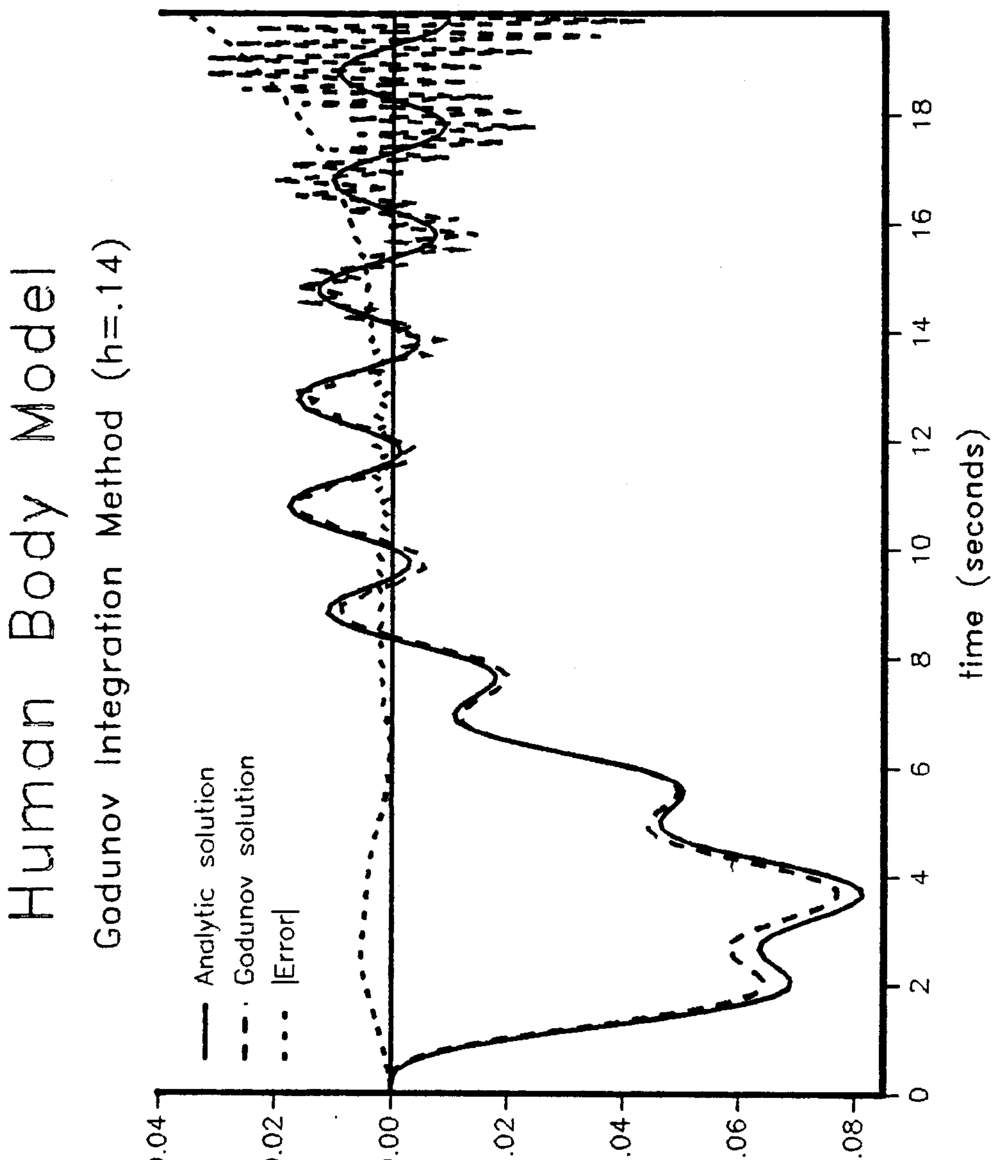


Figure 3.16 Human Body Model (h = .14)

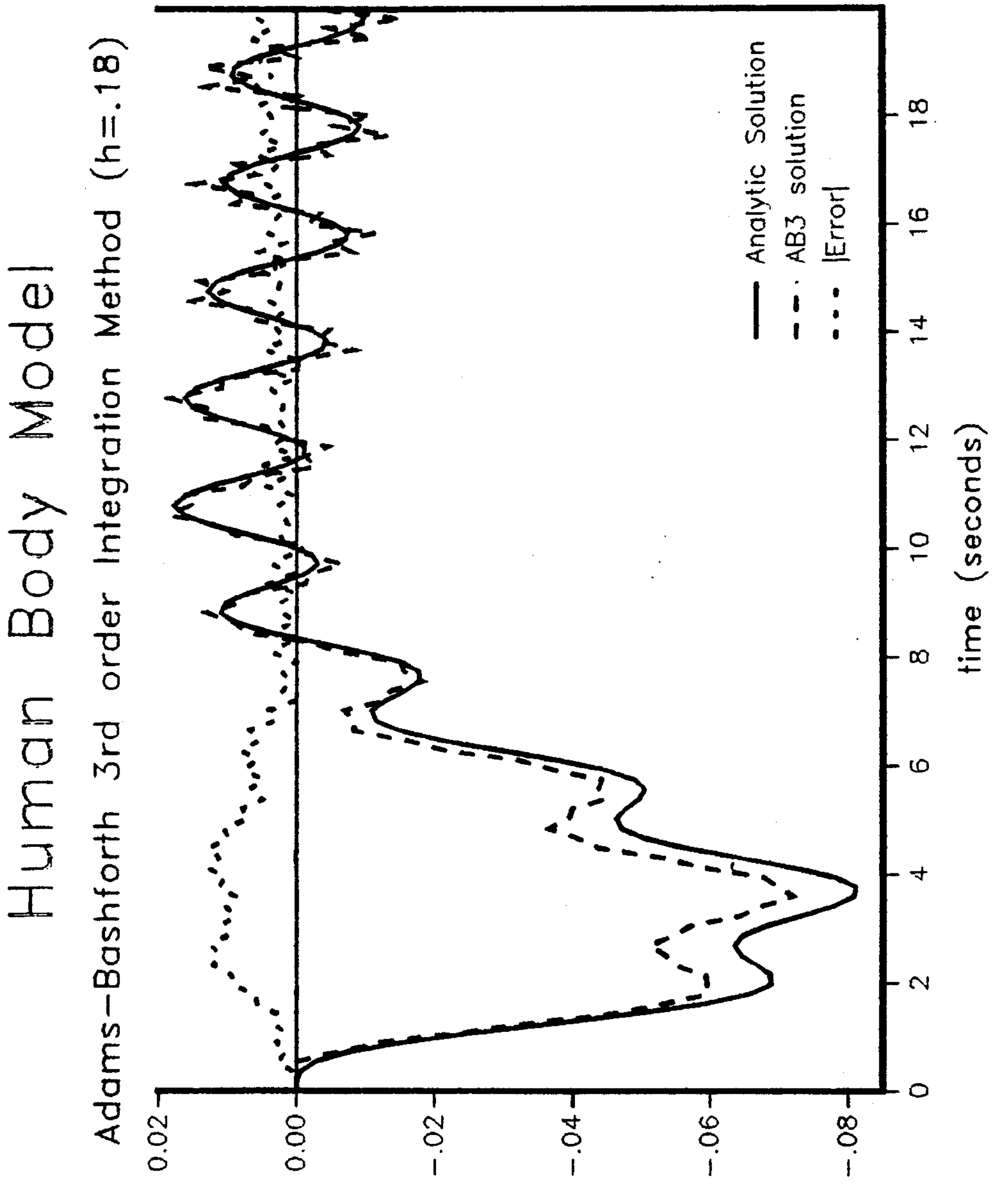


Figure 3.17 Human Body Model ($h = .18$)

THIRD EXAMPLE - PASSIVE ELECTRICAL SYSTEM

The passive electrical system example requires the solution of the equations

$$\begin{aligned} \ddot{\underline{x}}_1(t) &= A\underline{x}_1(t) + B\dot{\underline{x}}_1(t) + \underline{c}x_2(t) + \underline{d}\dot{u}(t) \\ \dot{x}_2(t) &= \underline{e}'\underline{x}_1(t) + fx_2(t) \end{aligned} \tag{3.6}$$

for a solution using the Godunov method where

$$A = \begin{pmatrix} -\frac{1}{L_1C_1} & \frac{1}{L_1C_1} & 0 & 0 \\ \frac{1}{L_1C_2} & -\frac{(L_1+L_2)}{C_2L_1L_2} & \frac{1}{L_2C_2} & 0 \\ 0 & \frac{1}{L_2C_3} & -\frac{1}{L_2C_3} & 0 \\ 0 & 0 & 0 & -\frac{1}{L_3C_4} \end{pmatrix}$$

$$B = \begin{pmatrix} -\frac{1}{R_1C_1} & 0 & 0 & 0 \\ 0 & -\frac{1}{R_2C_2} & 0 & 0 \\ 0 & 0 & -\frac{1}{R_3C_3} & \frac{1}{R_3C_3} \\ 0 & 0 & \frac{1}{R_3C_4} & -\frac{1}{R_3C_4} \end{pmatrix}$$

$$\underline{c} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{L_3C_4} \end{pmatrix}$$

$$\underline{d} = \begin{pmatrix} \frac{1}{C_1} \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\underline{e}' = \begin{pmatrix} 0 & 0 & 0 & \frac{R_4}{L_3} \end{pmatrix}$$

and $f = (-\frac{R_4}{L_3})$.

The first order differential equation solvers were given the system

$$\dot{\underline{x}}(t) = A\underline{x}(t) + \underline{b}u(t)$$

where

$$A = \begin{pmatrix} -\frac{1}{R_1C_1} & -\frac{1}{C_1} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{L_1} & 0 & -\frac{1}{L_1} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{C_2} & -\frac{1}{C_2R_2} & -\frac{1}{C_2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{L_2} & 0 & -\frac{1}{L_2} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{C_3} & -\frac{1}{C_3R_3} & \frac{1}{C_3R_3} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{C_4R_3} & -\frac{1}{C_4R_3} & -\frac{1}{C_4} \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{L_3} & -\frac{R_4}{L_3} \end{pmatrix}$$

$$\text{and} \quad \underline{b} = \begin{pmatrix} \frac{1}{C_1} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad .$$

The analytic solution to the electrical system,

$$\begin{aligned} v_5(t) = -5.0910^{-9}e^{-47.64t} &+ 2e^{-.4285t}(.3675cos2.0234t - .197sin2.0234t) \\ &+ 2e^{-.332t}(-.9161cos.9233t + 1.0621sin.9233t) \\ &+ 2e^{-.5455t}(.5834cos.3296t - 1.7868sin.3296t) \\ &+ (-.0348cos3t + .0597sin3t) \quad , \end{aligned}$$

is shown in Figure (3.18), and the analytic solution, the Godunov solution and the absolute value of the Godunov error are shown in Figure (3.19). The error increasing in a straight line indicates an instability in the numerical system which wasn't predicted in the stability analysis done in Chapter 2. At the same step size ($h = .01$) the Adams-Bashforth 3rd order method is much better behaved (Figure 3.20). The linearly increasing error of the Godunov solution was observed at step sizes approaching zero. Since the Godunov method seemed to work fine with the systems,

$$\ddot{\underline{x}}(t) = \underline{f}(\underline{x}(t), t) \tag{3.7}$$

$$\text{and} \quad \ddot{\underline{x}}(t) = \underline{f}(\underline{x}(t), \dot{\underline{x}}(t), u(t), t) \quad , \tag{3.8}$$

an attempt was made to modify equations (3.6) to fit the form of equation (3.8). This can be done by differentiating the first derivative equation of equations (2.51),

$$\dot{v}_5(t) = \frac{R_4}{L_3}v_4(t) - \frac{R_4}{L_3}v_5(t) \quad ,$$

to give

$$\ddot{v}_5(t) = \frac{R_4}{L_3}\dot{v}_4(t) - \frac{R_4}{L_3}\dot{v}_5(t) \quad .$$

The first four 2nd derivative equations remain unchanged. The system is now expressed in nearly the same form as that of the passive mechanical system except that the input $u(t)$ is differentiated.

$$\ddot{\underline{x}}(t) = \underline{f}(\underline{x}(t), \dot{\underline{x}}(t), \dot{u}(t), t) \quad . \tag{3.9}$$

Although this modified system is numerically stable, that is, the numerical solution does not approach infinity as t goes to infinity, the error between the numerical solution and the analytic solution is unacceptably large as shown in Figure (3.21). Another way of eliminating the equation

$$\dot{x}_2(t) = \underline{e}'\underline{x}_1(t) + fx_2(t) \tag{3.10}$$

in this case is to make $R_4 = 0$ (which of course isn't practical in the case of a real system for which we are attempting to arrive at a numerical solution). This eliminates the rightmost node of the circuit shown in Figure (2.3) which eliminates equation (3.10). This leaves, once again, a system in the form of equation (3.9). The Godunov method error for this system is bounded but still much larger in magnitude than the Adams-Bashforth 3rd order method error as seen in Figure (3.22).

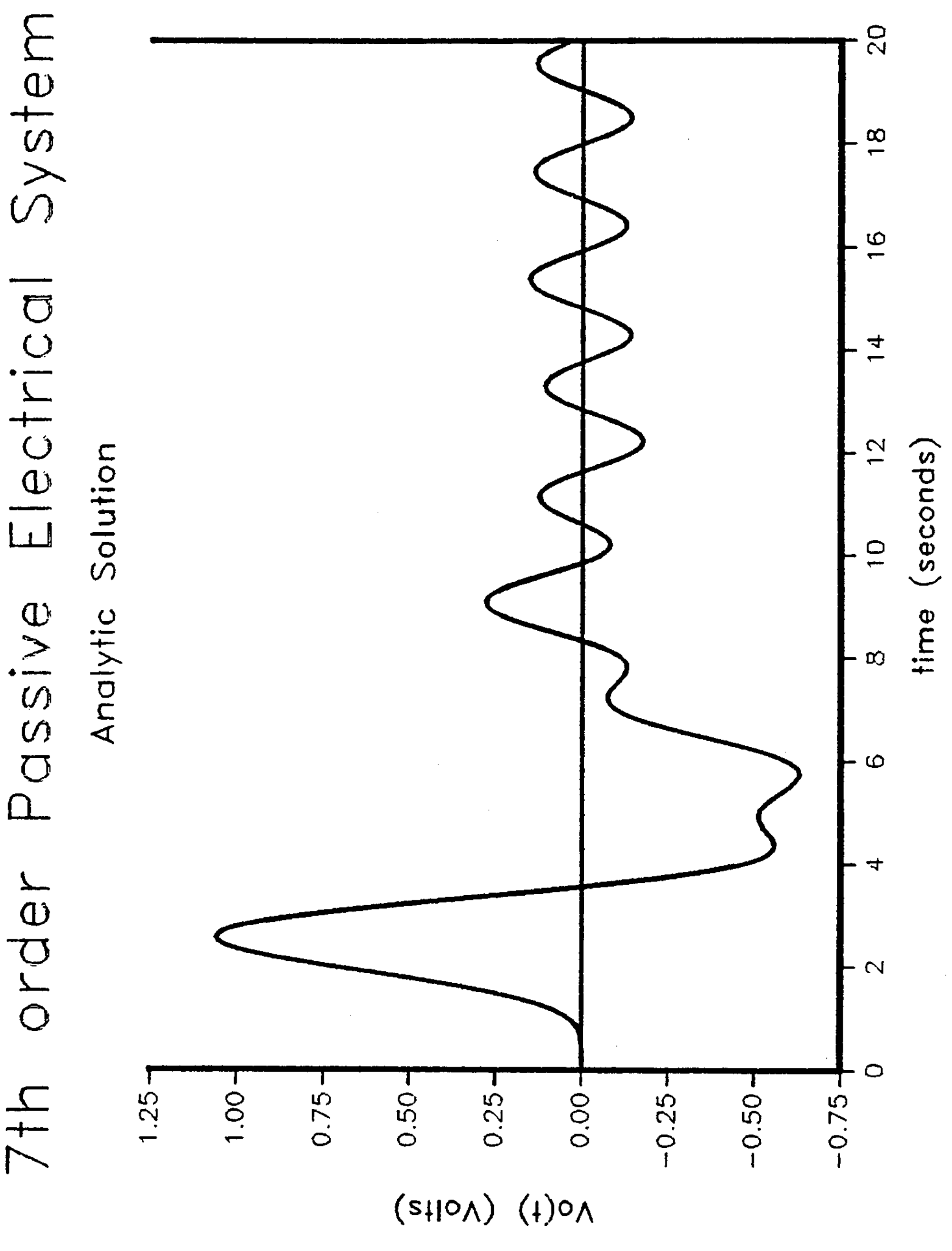


Figure 3.18 7th order Passive Electrical System (Analytic Solution)

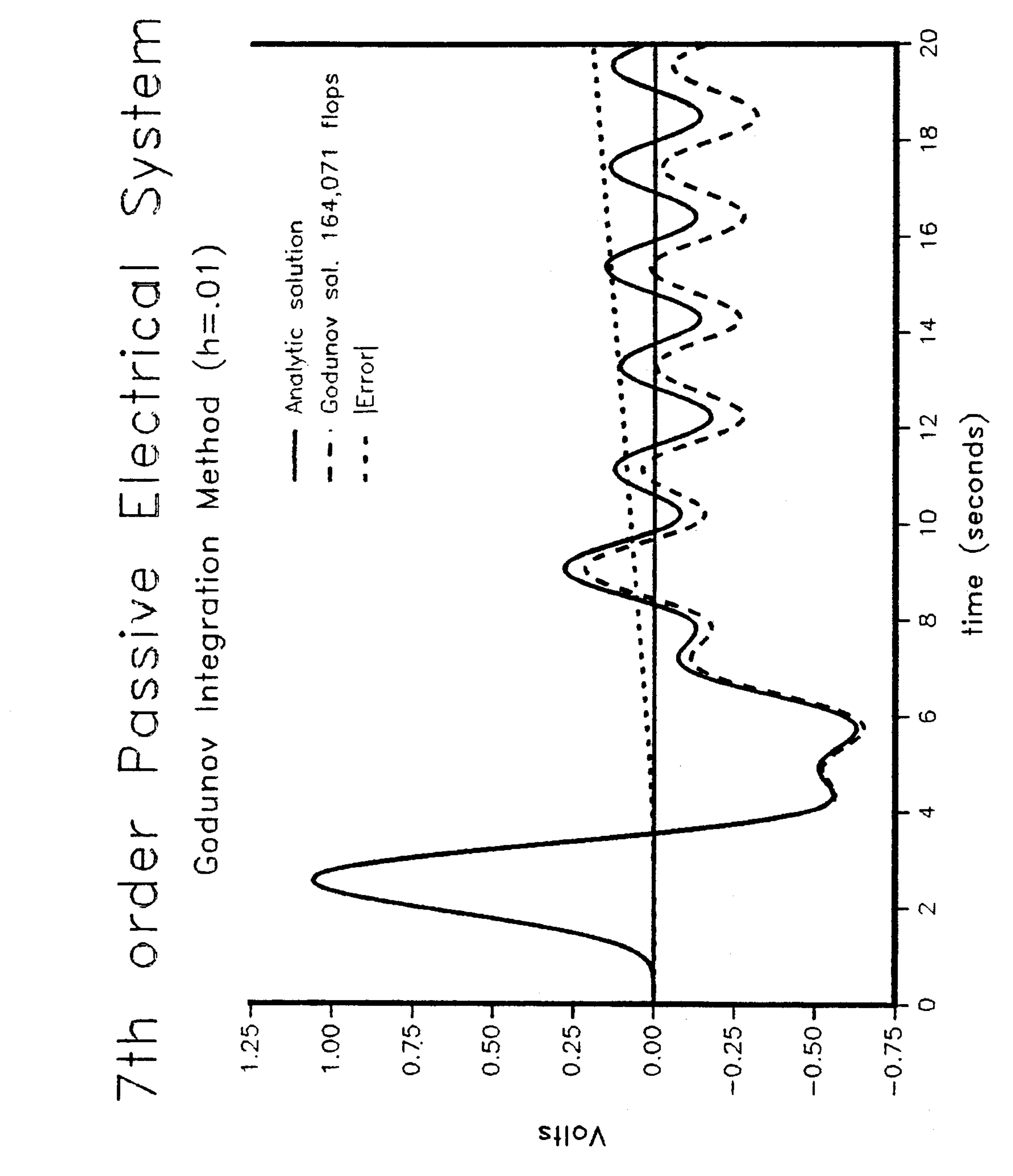


Figure 3.19 7th order Passive Electrical System ($h = .01$)

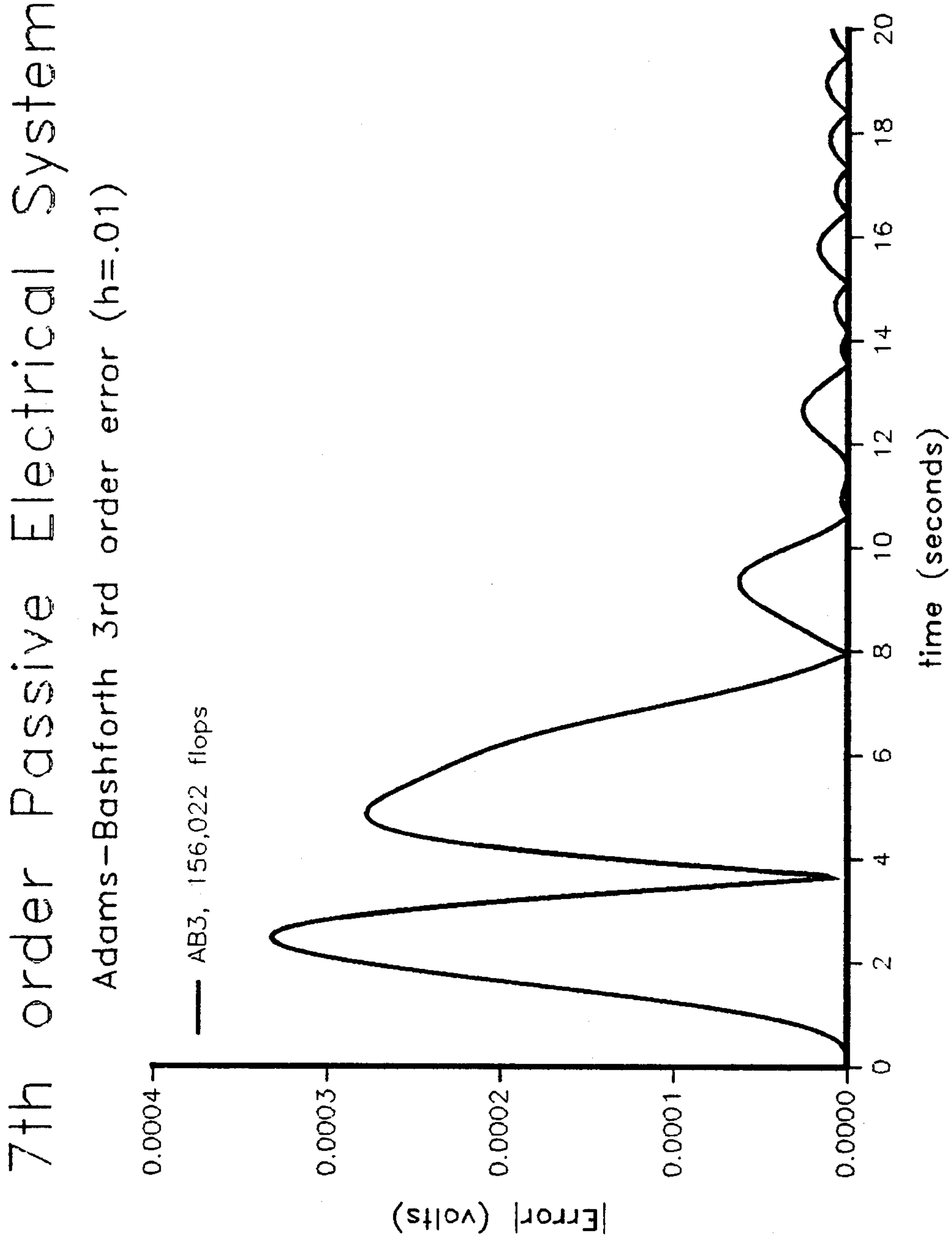


Figure 3.20 7th order Passive Electrical System ($h = .01$)

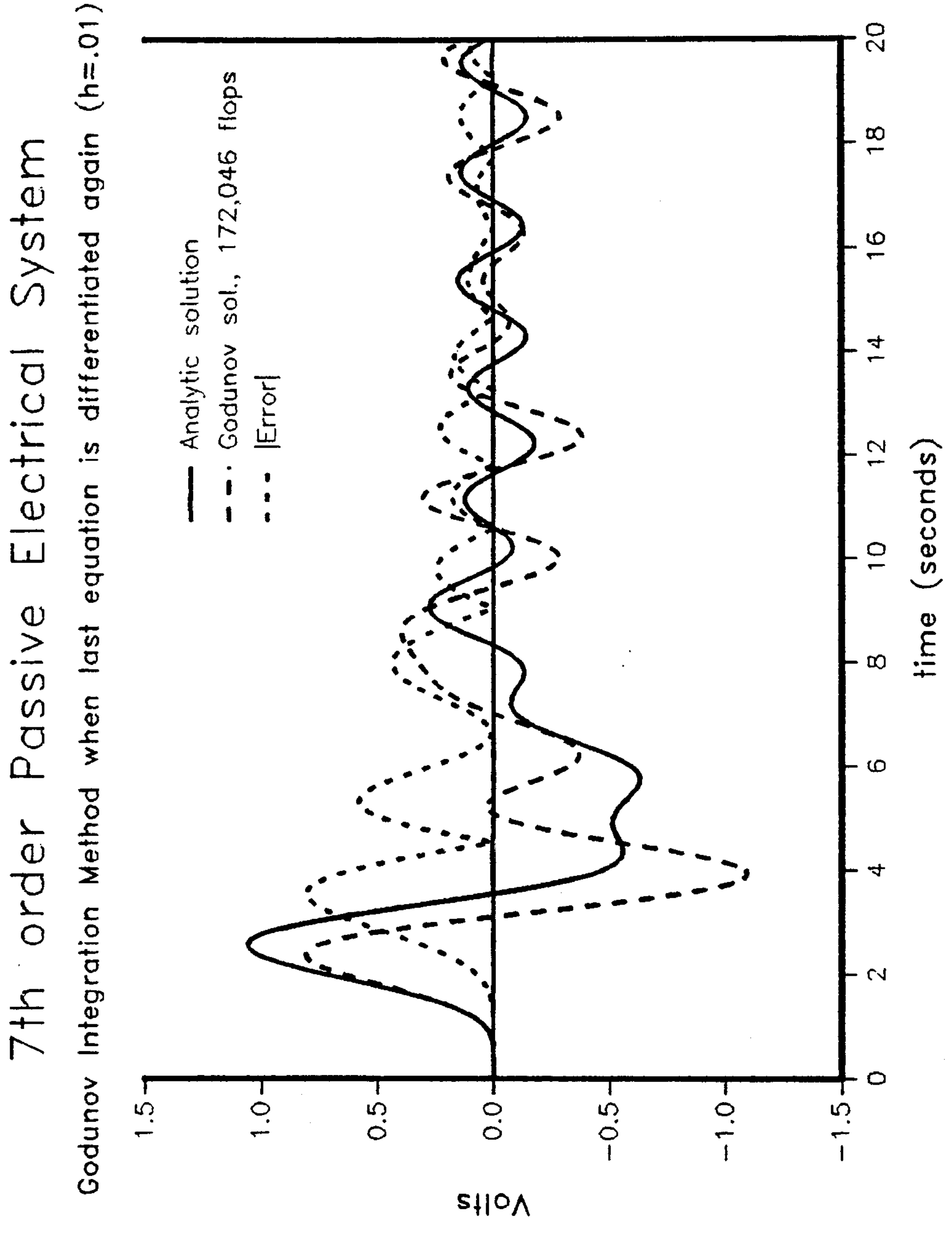


Figure 3.21 7th order Passive Electrical System ($h = .01$)

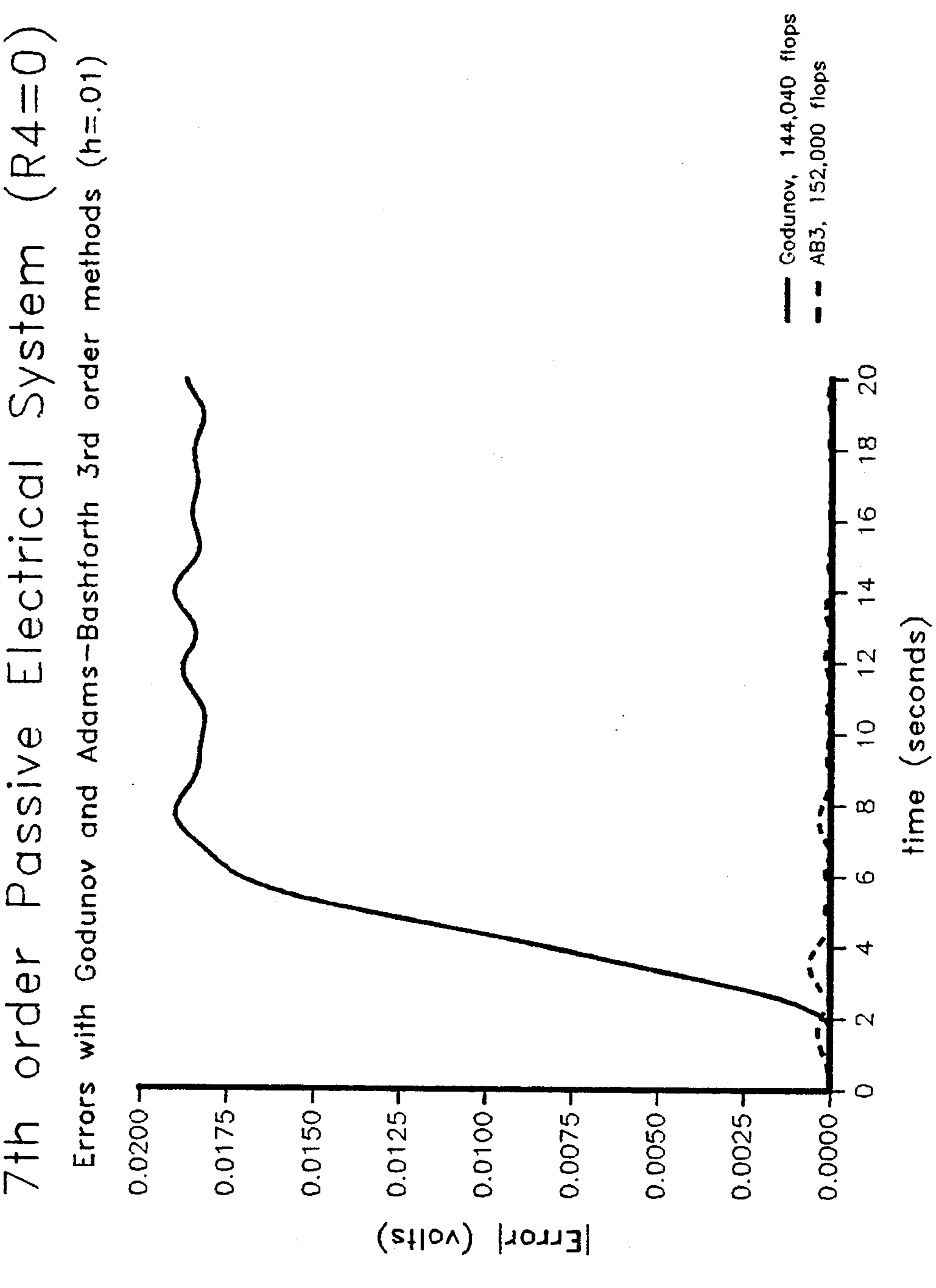


Figure 3.22 7th order Passive Electrical System ($R4 = 0, h = .01$)

In an attempt to determine under exactly what conditions the Godunov method does not arrive at a satisfactory solution, four small order electrical systems were solved numerically. The first system, shown in Figure (3.23) and referred to from here on as SYSTEM 1, results in the 2nd order differential equation

$$\ddot{x}(t) = -(\frac{1}{L_1C} + \frac{1}{L_2C})x(t) - \frac{1}{RC}\dot{x}(t) + \frac{1}{L_1C}u(t)$$

where

$$x(t) = v_o(t) \quad \text{and} \quad u(t) = v_{in}(t) = 10sin(3\pi t) \quad .$$

We will refer to this system as SYSTEM1G2, meaning SYSTEM1, to be solved using the Godunov method for 2nd order differential equations. This equation is of the form

$$\ddot{x}(t) = f(x, \dot{x}, u, t)$$

which is the same form of equation as the passive mechanical system resulted in.

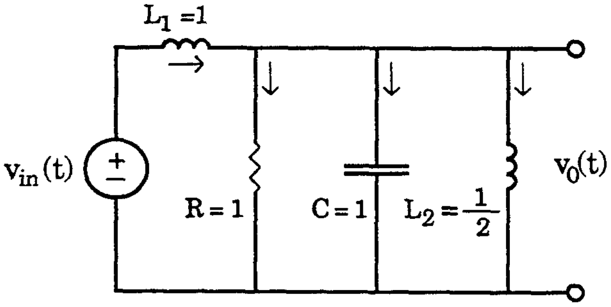


Figure 3.23 SYSTEM1

This equation can be separated into the two first order differential equations

$$\dot{x}_1(t) = x_2(t)$$
$$\dot{x}_2(t) = -(\frac{1}{L_1C} + \frac{1}{L_2C})x_1(t) - \frac{1}{RC}x_2(t) + \frac{1}{L_1C}u(t)$$

where

$$x_1(t) = v_o(t) \quad \text{and} \quad x_2(t) = \dot{v}_o(t) \quad .$$

This set of differential equations will be referred to as SYSTEM1A2, for SYSTEM1, Adams-Bashforth solution of the 2nd order differential equations. Another first order differential equation representation of this system can be arrived at by using $v_C(t)$ and $i_L(t)$ as the the state variables, which will result in the equations:

$$\begin{aligned} \dot{x}_1(t) &= \frac{1}{L_1}(u(t) - x_2(t)) \\ \dot{x}_2(t) &= \frac{1}{C}(x_1(t) - \frac{1}{R}x_2(t) - x_3(t)) \\ \dot{x}_3(t) &= \frac{1}{L_2}(x_2(t)) \end{aligned} \tag{3.11}$$

where

$$x_1(t) = i_{L_1}(t) \quad , x_2(t) = v_C(t) = v_o(t) \quad \text{and} \quad x_3(t) = i_{L_2}(t) \quad .$$

This set of differential equations will be referred to as SYSTEM1A1, for SYSTEM1, Adams-Bashforth solution of the 1st order differential equations.

The second small electrical system (Figure 3.24), which will be referred to as SYSTEM 2, resulted in two equations in the form of

$$\begin{aligned} \ddot{x}_1(t) &= f_1(x_1(t), \dot{x}_1(t), x_2(t), u(t), t) \\ \dot{x}_2(t) &= f_2(x_1(t), x_2(t), t) \quad . \end{aligned}$$

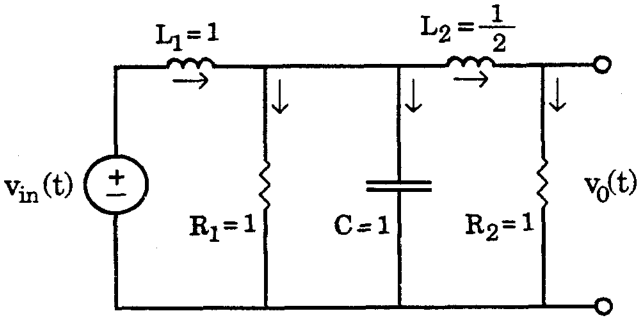


Figure 3.24 SYSTEM2

which has a first derivative equation but the input is not differentiated. The two equations resulting from this circuit are

$$\ddot{x}_1(t) = (-\frac{1}{L_1C} - \frac{1}{L_2C})x_1(t) - \frac{1}{R_1C}\dot{x}_1(t) + \frac{1}{L_2C}x_2(t) + \frac{1}{L_1C}u(t) \quad (3.12)$$

$$\dot{x}_2(t) = \frac{R_2}{L_2}x_1(t) - \frac{R_2}{L_2}x_2(t) \quad (3.13)$$

where

$$x_1(t) = v_C(t) \quad , \quad x_2(t) = v_o(t) \quad \text{and} \quad u(t) = v_{in}(t) = 10sin(3\pi t) \quad .$$

(Refer to these differential equations as SYSTEM2G2). If equation (3.12) is separated into two 1st order differential equations, the system can be expressed as

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= (-\frac{1}{L_1C} - \frac{1}{L_2C})x_1(t) - \frac{1}{R_1C}x_2(t) + \frac{1}{L_2C}x_3(t) + \frac{1}{L_1C}u(t) \\ \dot{x}_3(t) &= \frac{R_2}{L_2}x_1(t) - \frac{R_2}{L_2}x_3(t) \quad . \end{aligned}$$

where

$$x_1(t) = v_C(t) \quad , \quad x_2(t) = \dot{v}_C(t) \quad \text{and} \quad x_3(t) = v_o(t) \quad .$$

(Refer to this set of differential equations as SYSTEM2A2). Another formulation of the system equations in first order form is

$$\begin{aligned}\dot{x}_1(t) &= \frac{1}{L_1}(u(t) - x_2(t))\\ \dot{x}_2(t) &= \frac{1}{C}(x_1(t) - \frac{1}{R_1}x_2(t) - x_3(t))\\ \dot{x}_3(t) &= \frac{1}{L_2}(x_2(t) - R_2x_3(t)) \quad .\end{aligned}$$

where

$$x_1(t) = i_{L_1}(t) \quad , \quad x_2(t) = v_C(t) \quad \text{and} \quad x_3(t) = i_{L_2}(t) \quad .$$

(Refer to this set of differential equations as SYSTEM2A1). This time the system orders are reflected accurately in both the second order and the first order formulations. The third small electrical system to be experimented with is shown in Figure (3.25) and will be referred to as SYSTEM 3.

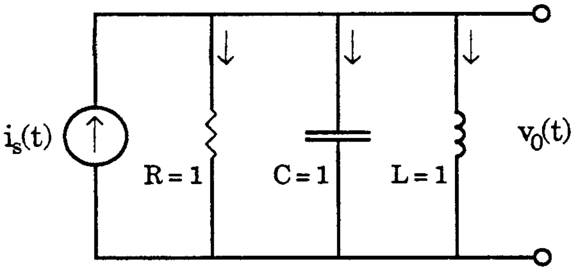


Figure 3.25 SYSTEM3

An equation of the form,

$$\ddot{x} = f(x(t), \dot{x}(t), \dot{u}(t), t) \quad ,$$

(which requires the derivative of the input but not a first derivative equation) results from this system:

$$\ddot{x}(t) = -\frac{1}{LC}x(t) - \frac{1}{RC}\dot{x}(t) + \dot{u}(t) \tag{3.14}$$

where

$$x(t) = v_C(t) \quad \text{and} \quad u(t) = i_s(t) = 10sin(3\pi t) \quad .$$

(Refer to this differential equation as SYSTEM3G2). Equation (3.14), when separated into two first order differential equations, results in:

$$\begin{aligned}\dot{x}_1(t) &= x_2(t)\\ \dot{x}_2(t) &= -\frac{1}{LC}x_1(t) - \frac{1}{RC}x_2(t) + \dot{u}(t)\end{aligned}$$

where

$$x_1(t) = v_C(t) \quad \text{and} \quad x_2(t) = \dot{v}_C(t) \quad .$$

(This set of differential equations will be referred to as SYSTEM3A2). This system can also be formulated in the form of the two 1st order differential equations:

$$\begin{aligned}\dot{x}_1(t) &= \frac{1}{C}(u(t) - \frac{1}{R}x_1(t) - x_2(t))\\ \dot{x}_2(t) &= \frac{1}{L}(x_1(t))\end{aligned}$$

where

$$x_1(t) = v_C(t) \quad \text{and} \quad x_2(t) = i_L(t) \quad .$$

(This set of differential equations will be referred to as SYSTEM3A1). The last small electrical system used for this experiment (Figure 3.26), referred to as SYSTEM 4, results in equations of the form

$$\begin{aligned}\ddot{x}_1(t) &= f_1(x_1(t), \dot{x}_1(t), x_2(t), \dot{u}(t), t)\\ \dot{x}_2(t) &= f_2(x_1(t), x_2(t), t) \quad ,\end{aligned}$$

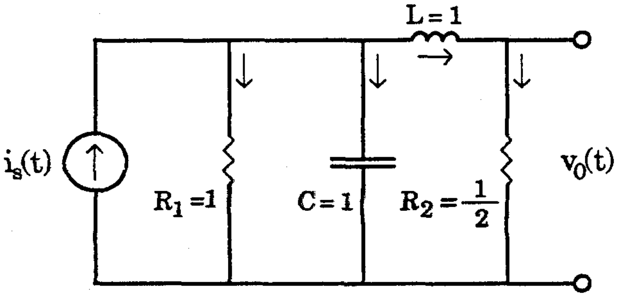


Figure 3.26 SYSTEM4

which have both the first derivative equation and a differentiated input. The equations

$$\ddot{x}_1(t) = -\frac{1}{LC}x_1(t) - \frac{1}{R_1C}\dot{x}_1(t) + \frac{1}{LC}x_2(t) + \frac{1}{C}\dot{u}(t) \tag{3.15}$$

$$\dot{x}_2(t) = \frac{R_2}{L}x_1(t) - \frac{R_2}{L}x_2(t) \tag{3.16}$$

result from this circuit, with

$$x_1(t) = v_C(t) \quad , \quad x_2(t) = v_o(t) \quad \text{and} \quad u(t) = i_s(t) = 10sin(3\pi t) \quad .$$

(Refer to these differential equations as SYSTEM4G2). Equation (3.15) can also be separated into two 1st order differential equations so that

$$\begin{aligned}
\dot{x}_1(t) &= x_2(t) \\
\dot{x}_2(t) &= -\frac{1}{LC}x_1(t) - \frac{1}{R_1C}x_2(t) + \frac{1}{LC}x_3(t) + \frac{1}{C}\dot{u}(t) \\
\dot{x}_3(t) &= \frac{R_2}{L}x_1(t) - \frac{R_2}{L}x_3(t)
\end{aligned}$$

where

$$x_1(t) = v_C(t) \quad , \quad x_2(t) = \dot{v}_C(t) \quad \text{and} \quad x_3(t) = v_o(t) \quad .$$

(Refer to these differential equations as SYSTEM4A2). The second formulation of system equations in 1st order form is

$$\begin{aligned}\dot{x}_1(t) &= \frac{1}{C}(u(t) - \frac{1}{R_1}x_1(t) - x_2(t))\\ \dot{x}_2(t) &= \frac{1}{L}(x_1(t) - R_2x_2(t)) \quad .\end{aligned}$$

(Refer to these differential equations as SYSTEM4A1). In this example, the second order equation formulation of the system equations makes a second order system appear to be a third order system.

The analytic solutions, and the error resulting from the numerical solutions of these 4 small electrical systems are shown in figures (3.27) through (3.35). It can be seen that the Godunov solution remains stable for the three system types:

$$\ddot{x}(t) = f(x(t), \dot{x}(t), u(t), t) \quad ,$$

$$\begin{aligned}\ddot{x}_1(t) &= f_1(x_1(t), \dot{x}_1(t), x_2(t), u(t), t)\\ \dot{x}_2(t) &= f_2(x_1(t), x_2(t), t) \quad ,\end{aligned}$$

$$\text{and} \quad \ddot{x}(t) = f(x(t), \dot{x}(t), \dot{u}(t), t) \quad ,$$

although the errors resulting from the Godunov solution of these three equation types is significantly higher than the Adams-Bashforth 3rd order errors. The system type:

$$\begin{aligned}\ddot{x}_1(t) &= f_1(x_1(t), \dot{x}_1(t), x_2(t), \dot{u}(t), t)\\ \dot{x}_2(t) &= f_2(x_1(t), x_2(t), t) \quad ,\end{aligned} \tag{3.17}$$

however, results in an unstable solution for the Godunov method.

A deeper understanding of the reason for the unstable Godunov solution to the system type of (3.17) can be found by rexamining the stability analysis of the passive mechanical and electrical systems of Chapter 2. If we substitute the matrices and vectors, A through $\underline{e}'$, of SYSTEM4G2 and $h = 0.05$ into equations (2.55) and put these equations into the form $\underline{x}_n = \phi^n \underline{x}_0$, we find the eigenvalues of ϕ to be:

$eig(\phi)$	$\lvert eig(\phi)\rvert$
0.0000-0.0000i	0.0000
0.0924-0.0000i	0.0924
-0.1156-0.0000i	0.1156
0.1263-0.0000i	0.1263
-0.1734-0.0000i	0.1734
0.9621+0.0466i	0.9632
0.9621-0.0466i	0.9632
1.0000-0.0000i	1.0000
1.0000+0.0000i	1.0000

Table 3.1 Eigenvalues of SYSTEM4 discretized with hybrid numerical method

If the same technique is used for the Adams-Bashforth solution of SYSTEM4A2, we find the ϕ matrix to have eigenvalues of:

$eig(\phi)$	$\lvert eig(\phi)\rvert$
0.0000-0.0000i	0.0000
0.0000+0.0000i	0.0000
0.1278+0.0545i	0.1390
0.1278-0.0545i	0.1390
-0.1618+0.1007i	0.1906
-0.1618-0.1007i	0.1906
0.9621-0.0466i	0.9632
0.9621+0.0466i	0.9632
1.0000-0.0000i	1.0000

Table 3.2 Eigenvalues of SYSTEM4 discretized with AB3 method

The eigenvalues of the discrete system which are at $z = 1$ correspond to eigenvalues in the continuous domain of $s = 0$, or free integrators. This explains why SYSTEM4A2, with one free integrator, shows a step output; and why SYSTEM4G2, which has two free integrators, shows a ramp in the output. Similar results are obtained by using this analysis with different stepsizes and also with the *7th* order passive electrical system.

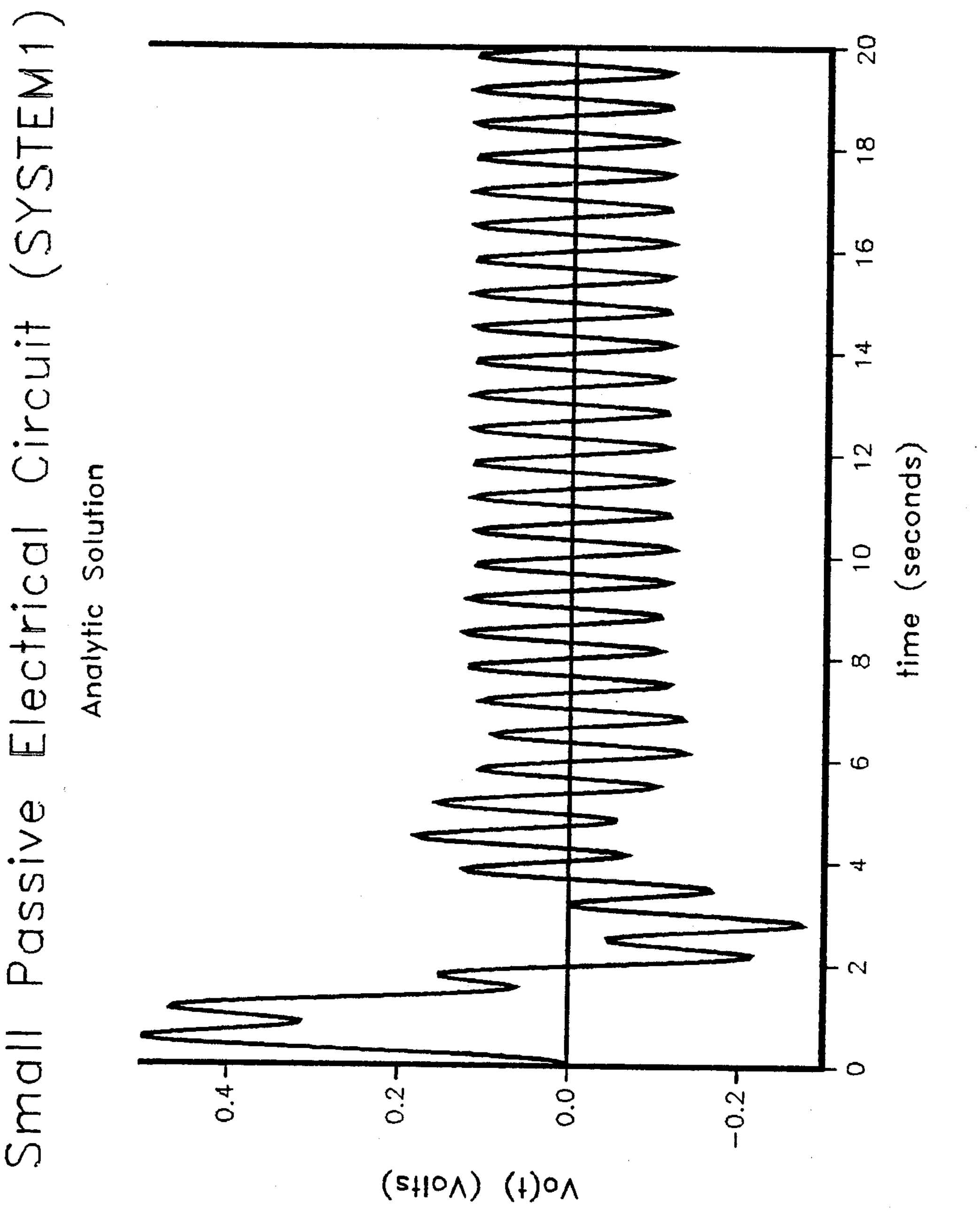


Figure 3.27 SYSTEM1 (Analytic Solution)

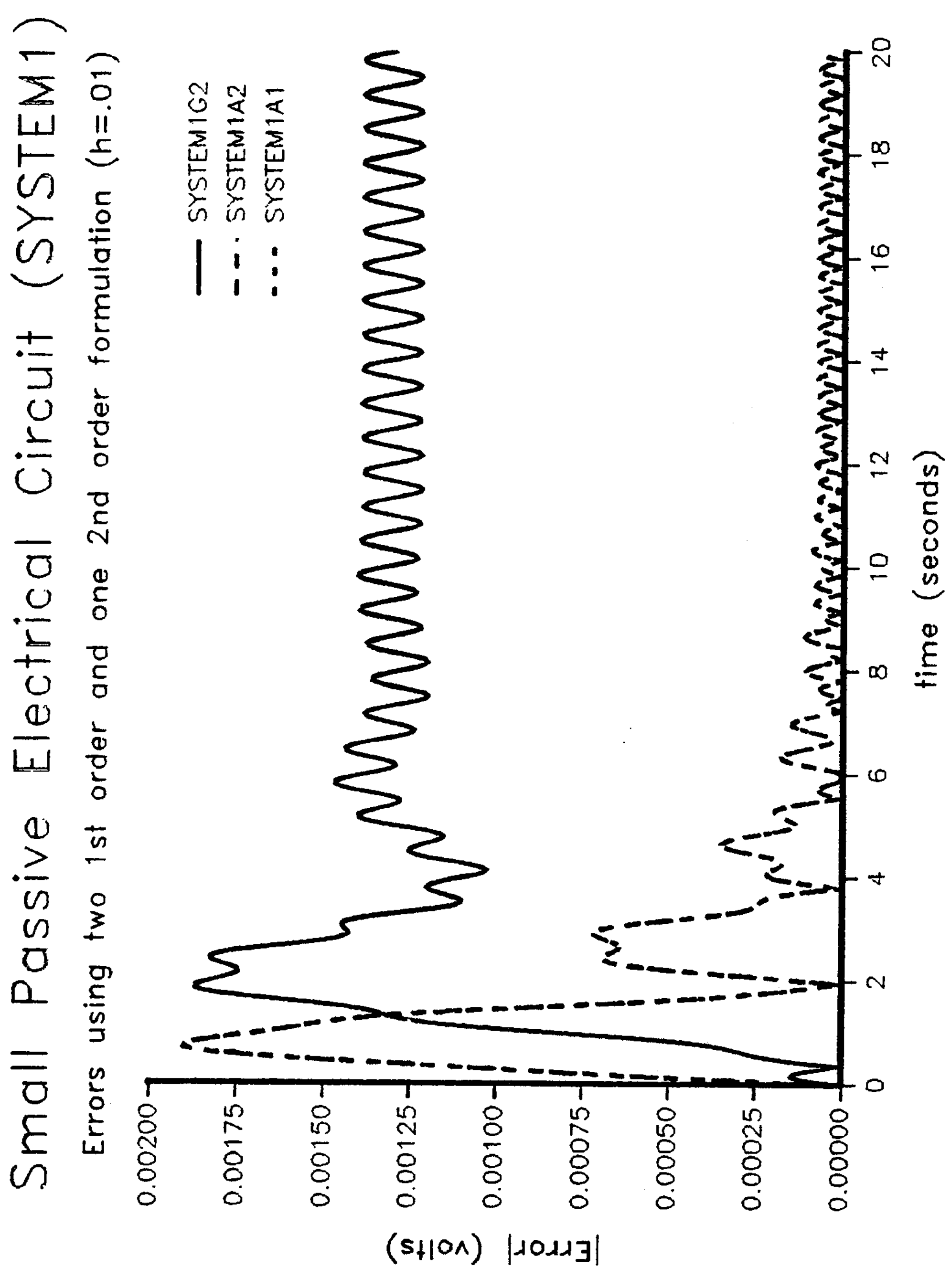


Figure 3.28 SYSTEM1 (h = .01)

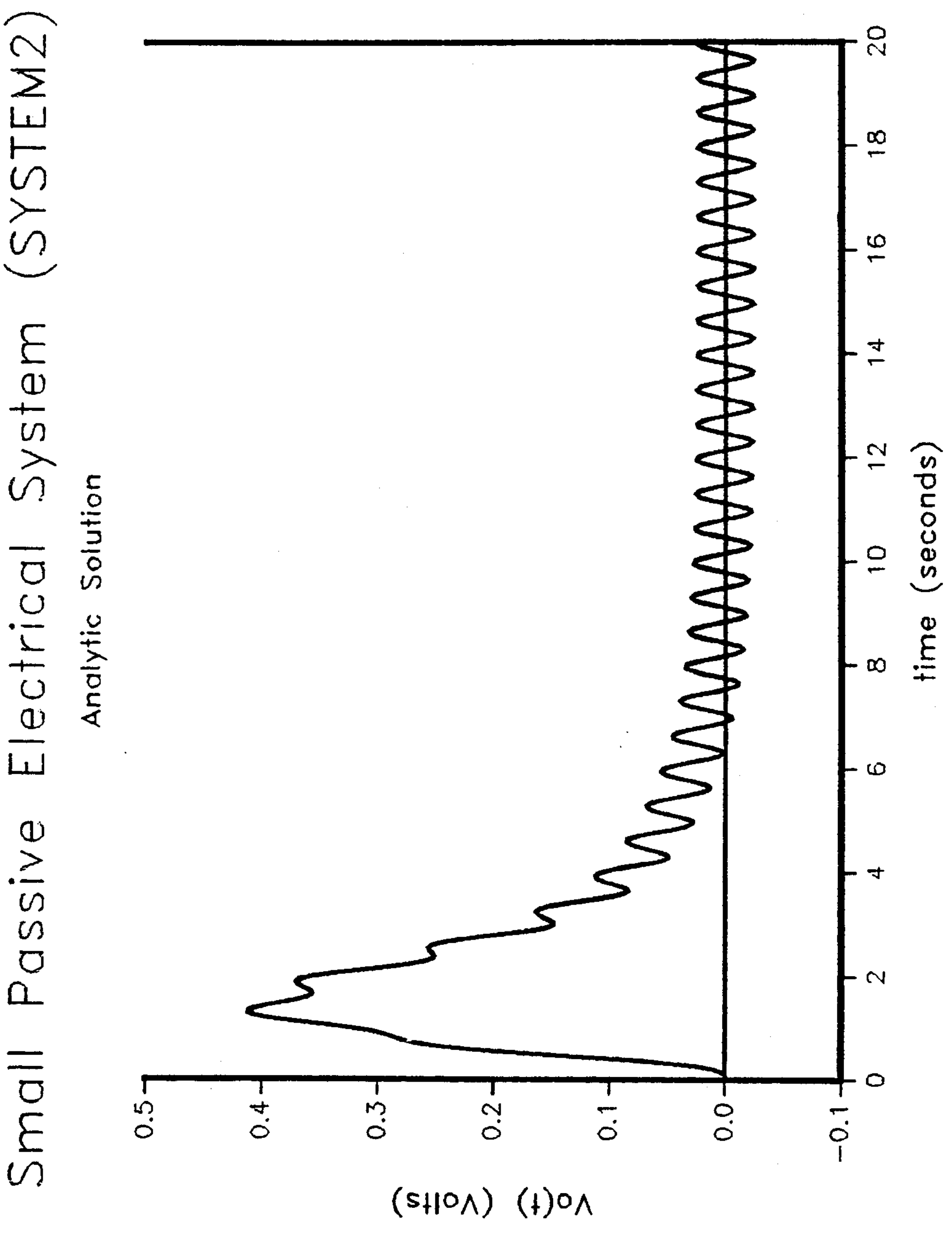


Figure 3.29 SYSTEM2 (Analytic Solution)

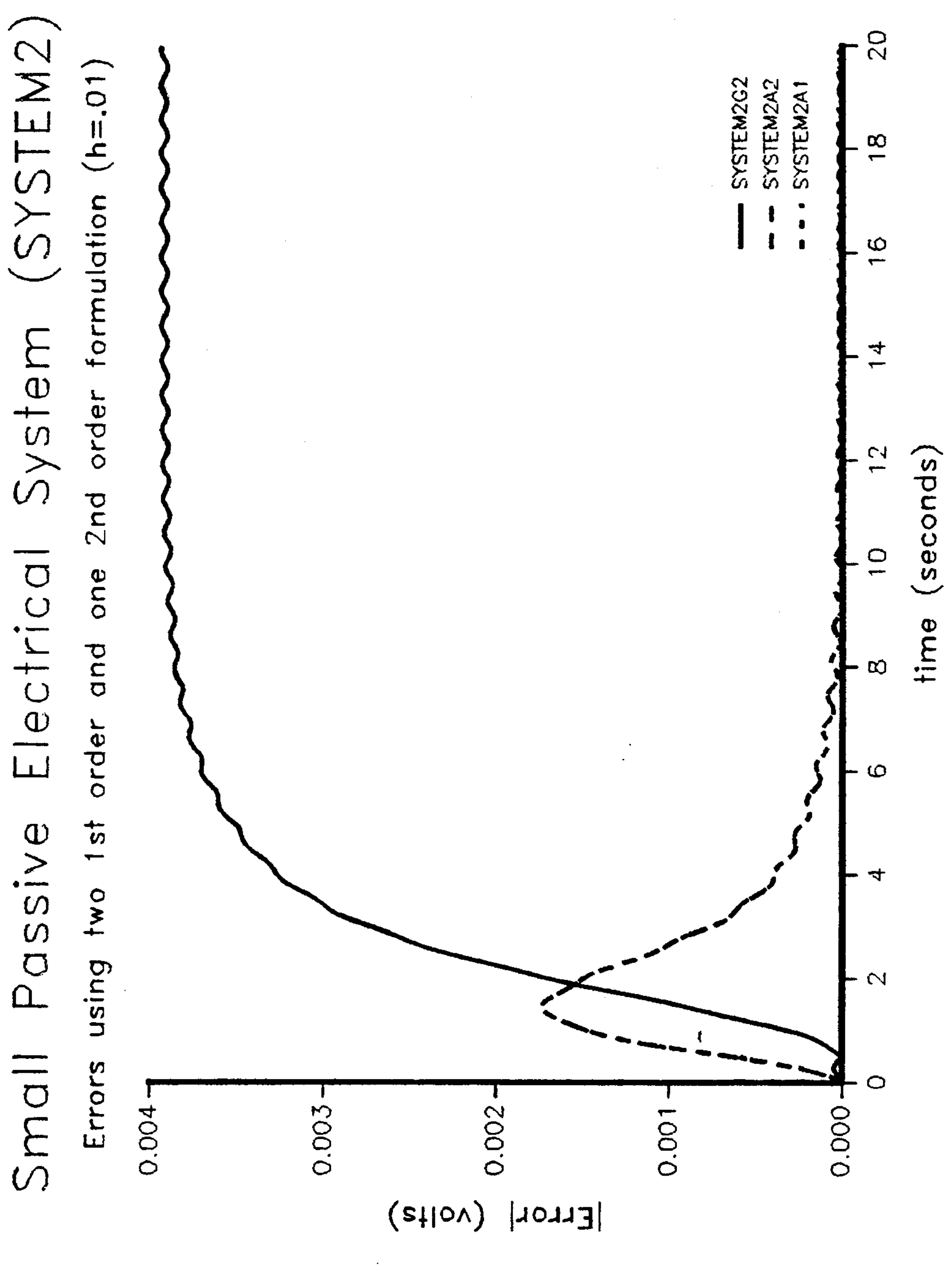


Figure 3.30 SYSTEM2 (h = .01)

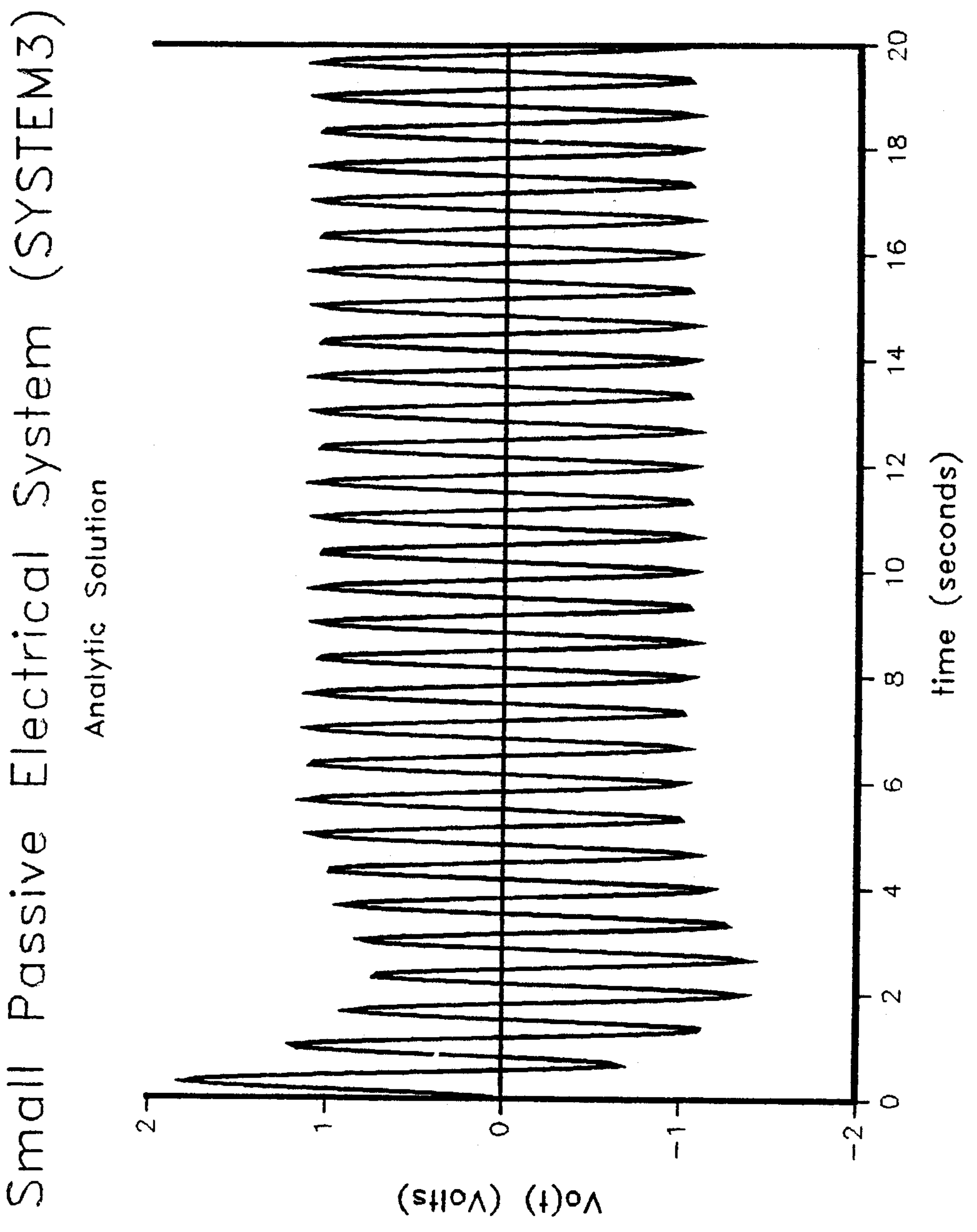


Figure 3.31 SYSTEM3 (Analytic Solution)

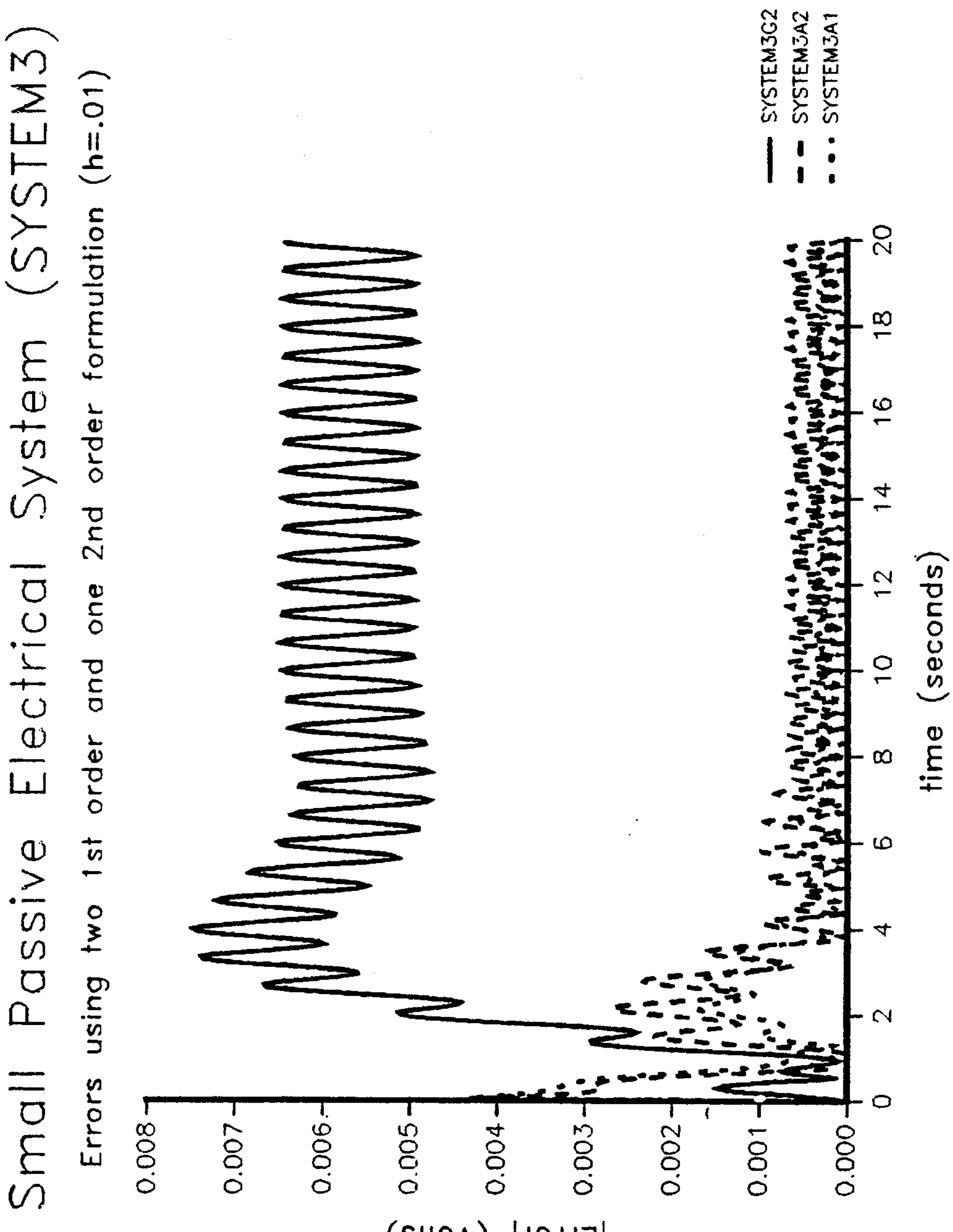


Figure 3.32 SYSTEM3 ($h = .01$)

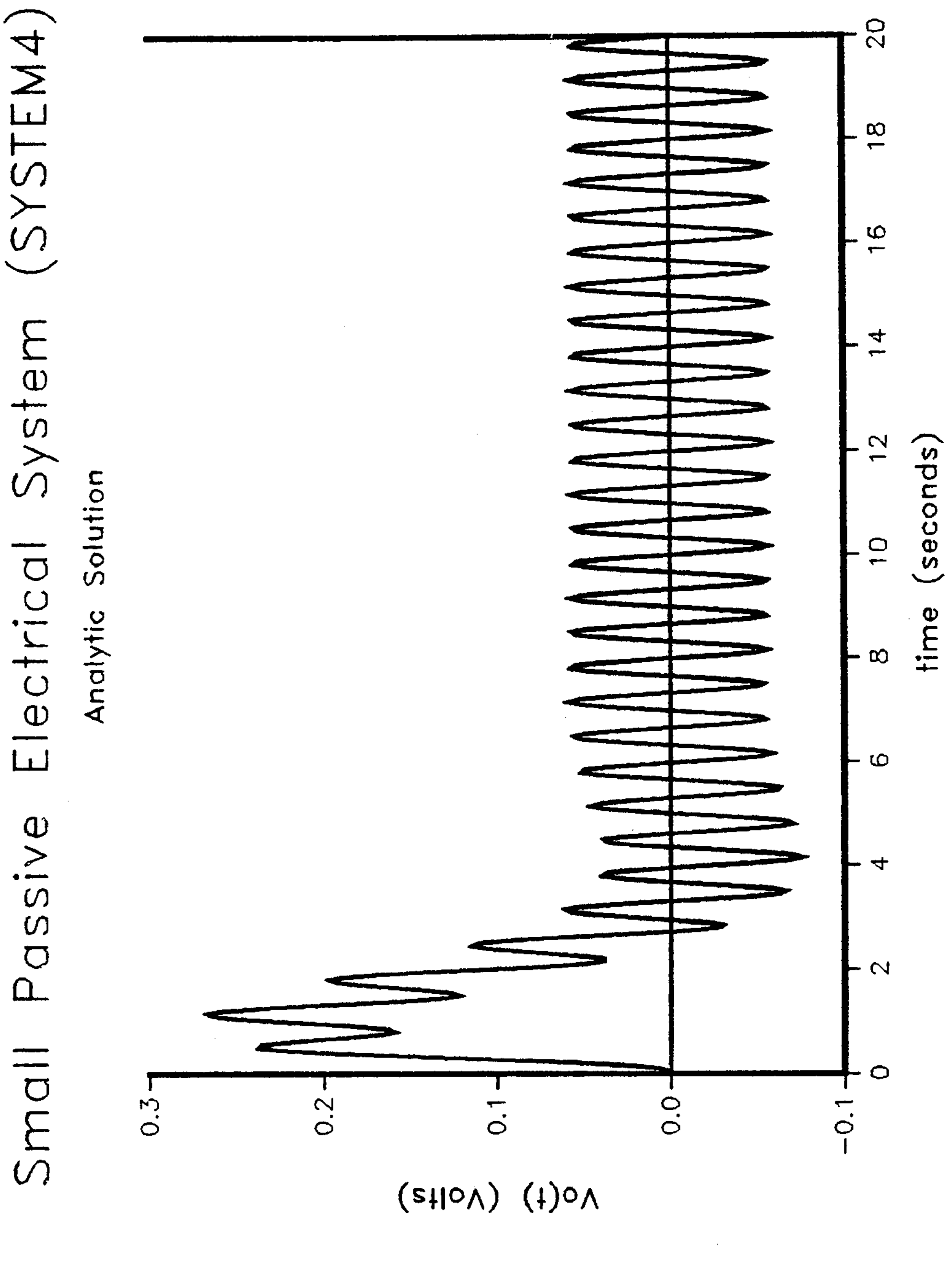


Figure 3.33 SYSTEM4 (Analytic Solution)

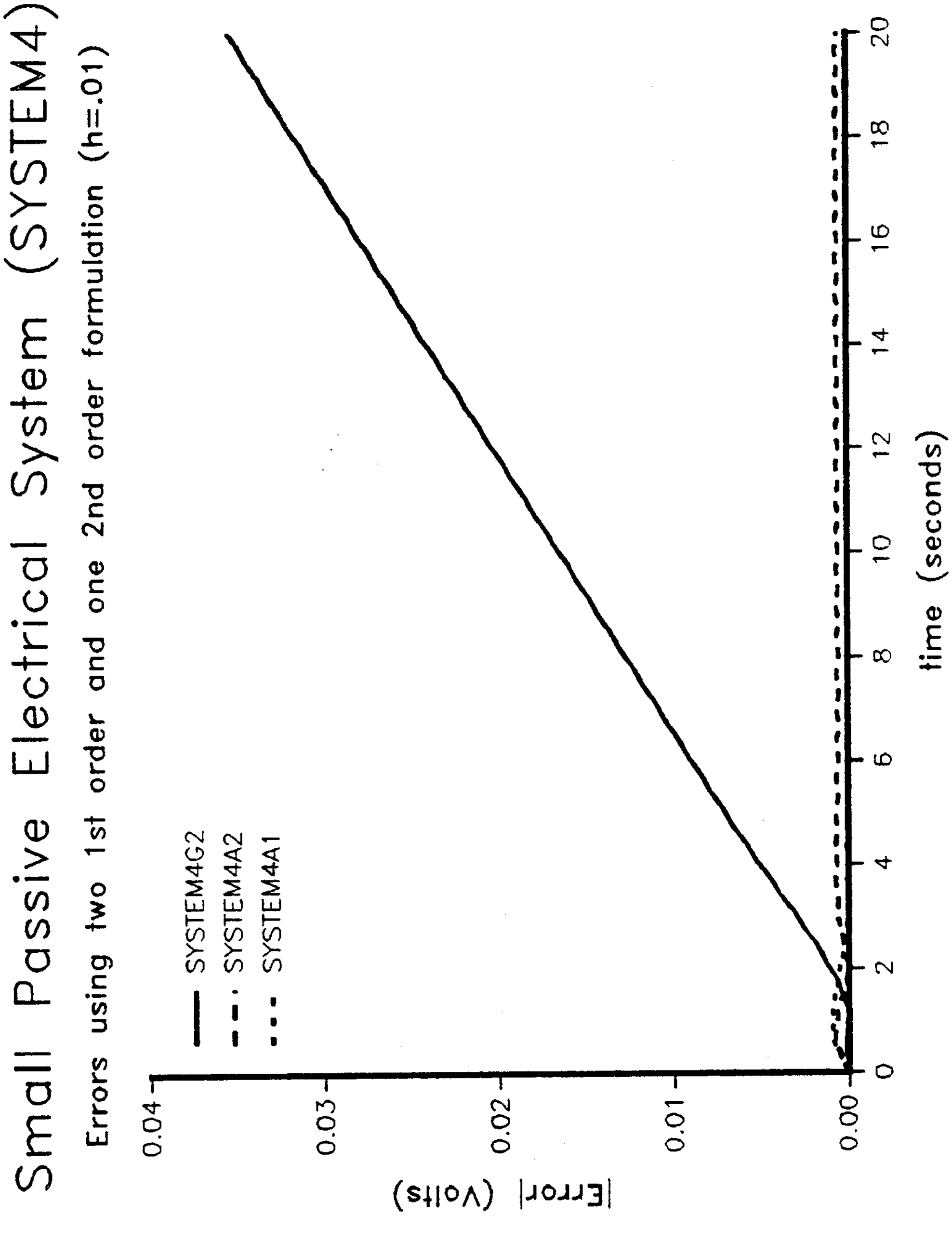


Figure 3.34 SYSTEM4 (h = .01)

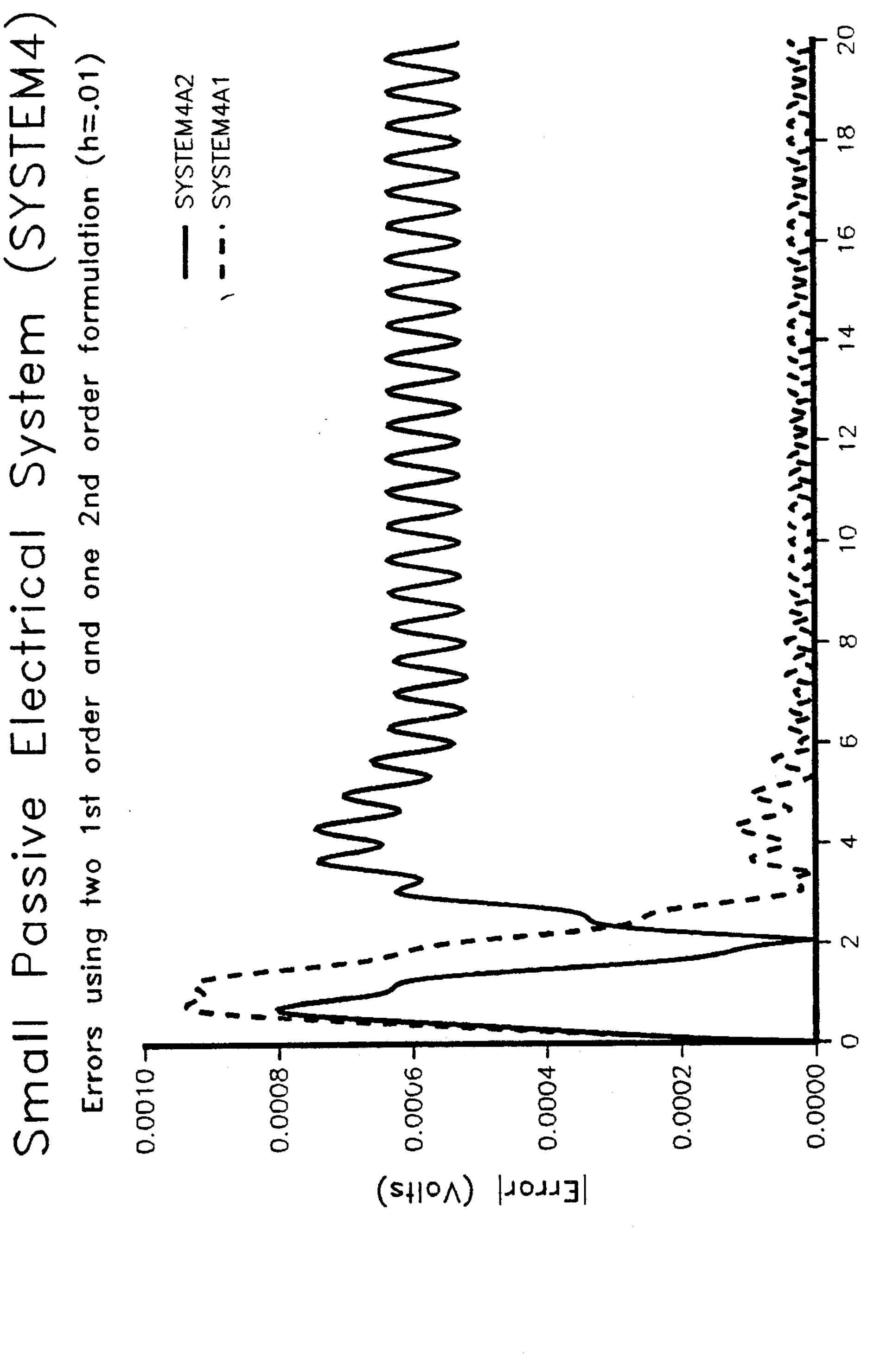


Figure 3.35 SYSTEM4 ($h = .01$)

FOURTH EXAMPLE - SINE GORDON EQUATION

The analytic solution to the sine-Gordon equation,

$$\frac{\partial^2 \phi}{\partial t^2} = \frac{\partial^2 \phi}{\partial x^2} - sin\phi \tag{3.18}$$

with initial conditions

$$\phi(x,0) = 4tan^{-1}[e^{(\frac{x}{\sqrt{1-u^2}})}] \quad ,$$

$$\text{and} \quad \frac{\partial \phi(x,0)}{\partial t} = \frac{-4ue^{(\frac{x}{\sqrt{1-u^2}})}}{(1+e^{(\frac{2x}{\sqrt{1-u^2}})})(\sqrt{1-u^2})} \quad .$$

is

$$\phi(x,t) = 4tan^{-1}[e^{\pm(\frac{x-ut}{\sqrt{1-u^2}})}]$$

for $x = -\infty$ to ∞. The analytic solution from $x = -10$ to 10 with $u = \frac{1}{2}$ is shown in Figure (3.36). The errors resulting from the numerical solution of this problem using the Godunov method, Adams-Bashforth 3rd order method and the Runge-Kutta 4th order method using a step size of $h = 0.01$ are shown in Figures (3.37) through (3.39). As was seen in the numerical solution of the wave equation with $h = 0.01$, the Godunov method takes approximately half as many floating point operations as does the Adams-Bashforth 3rd order method, and the discretization errors for the two methods are virtually identical. This is not unexpected since the partial differential equations of the two systems are very similar.

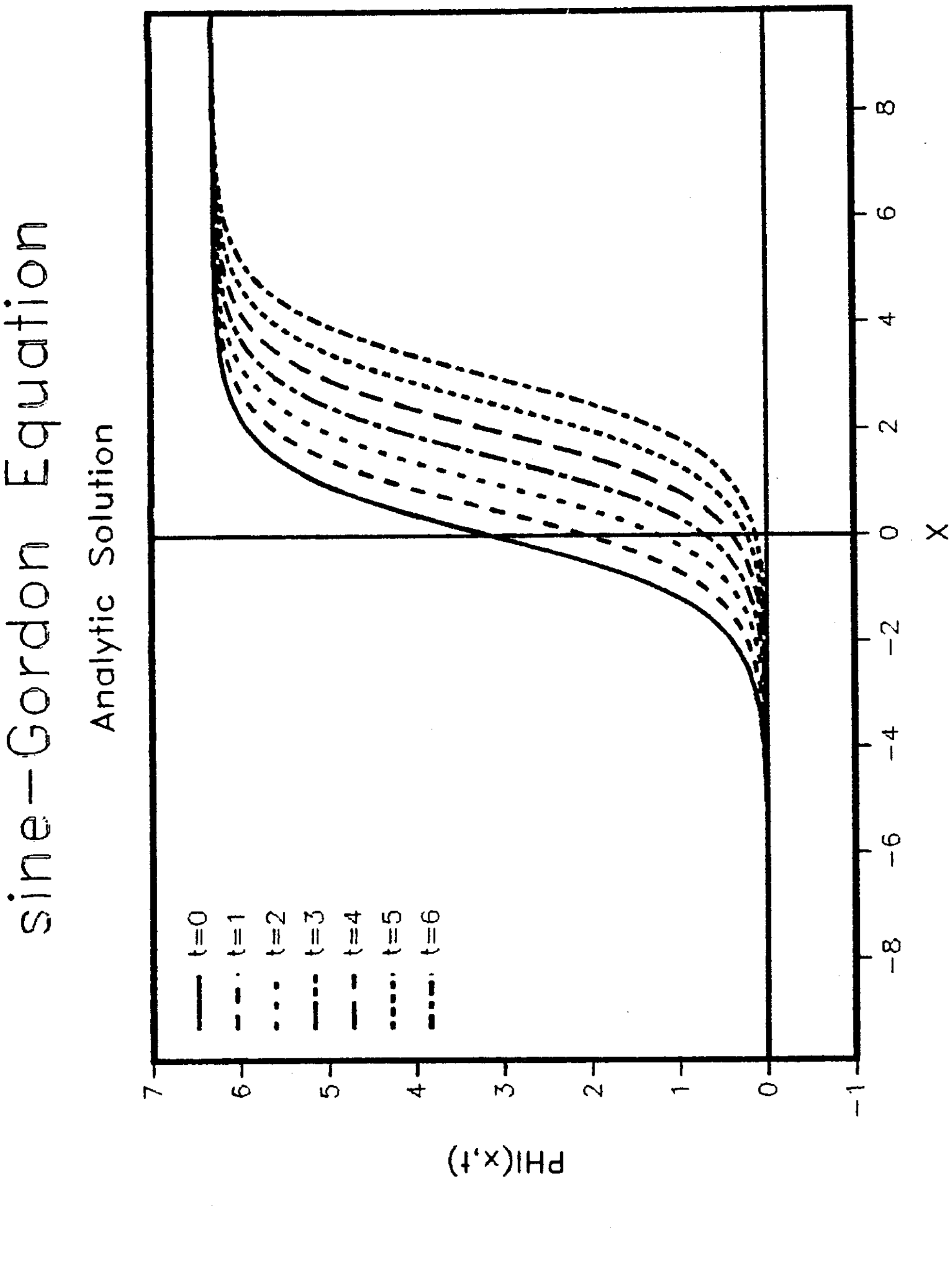


Figure 3.36 sine-Gordon Equation (Analytic Solution)

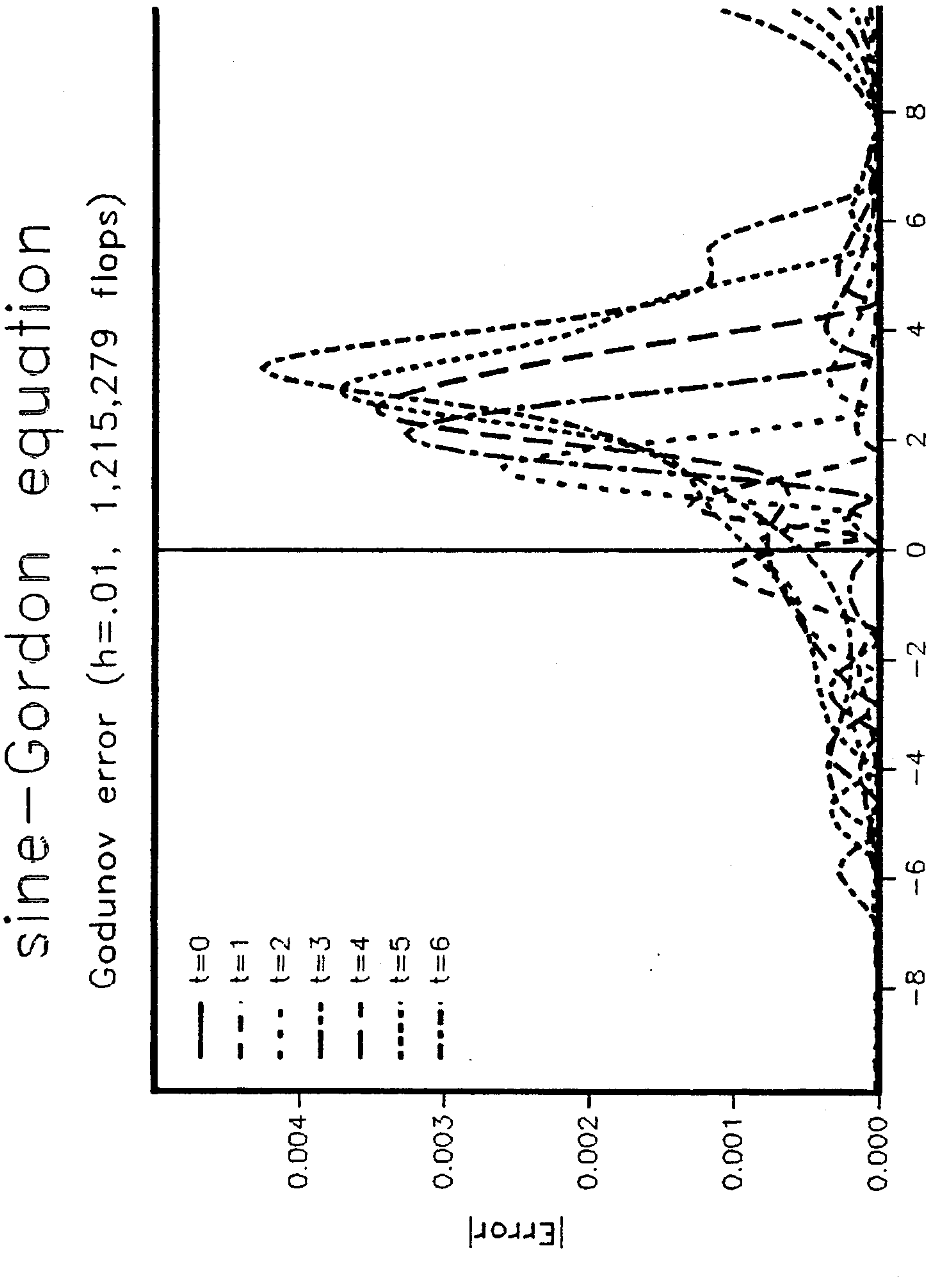


Figure 3.37 sine-Gordon Equation (Godunov Error)

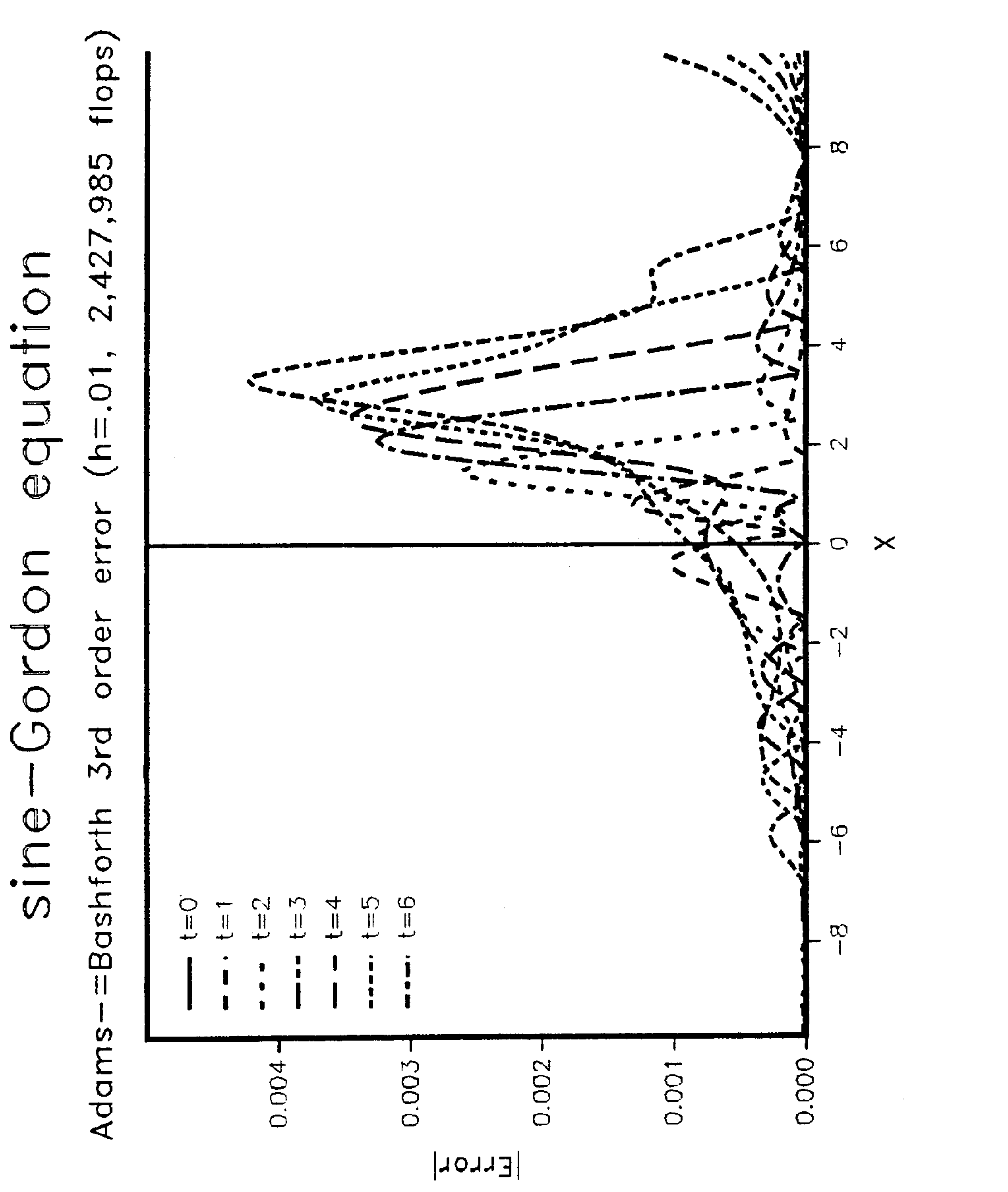


Figure 3.38 sine-Gordon Equation (Adams-Bashforth 3rd Order Error)

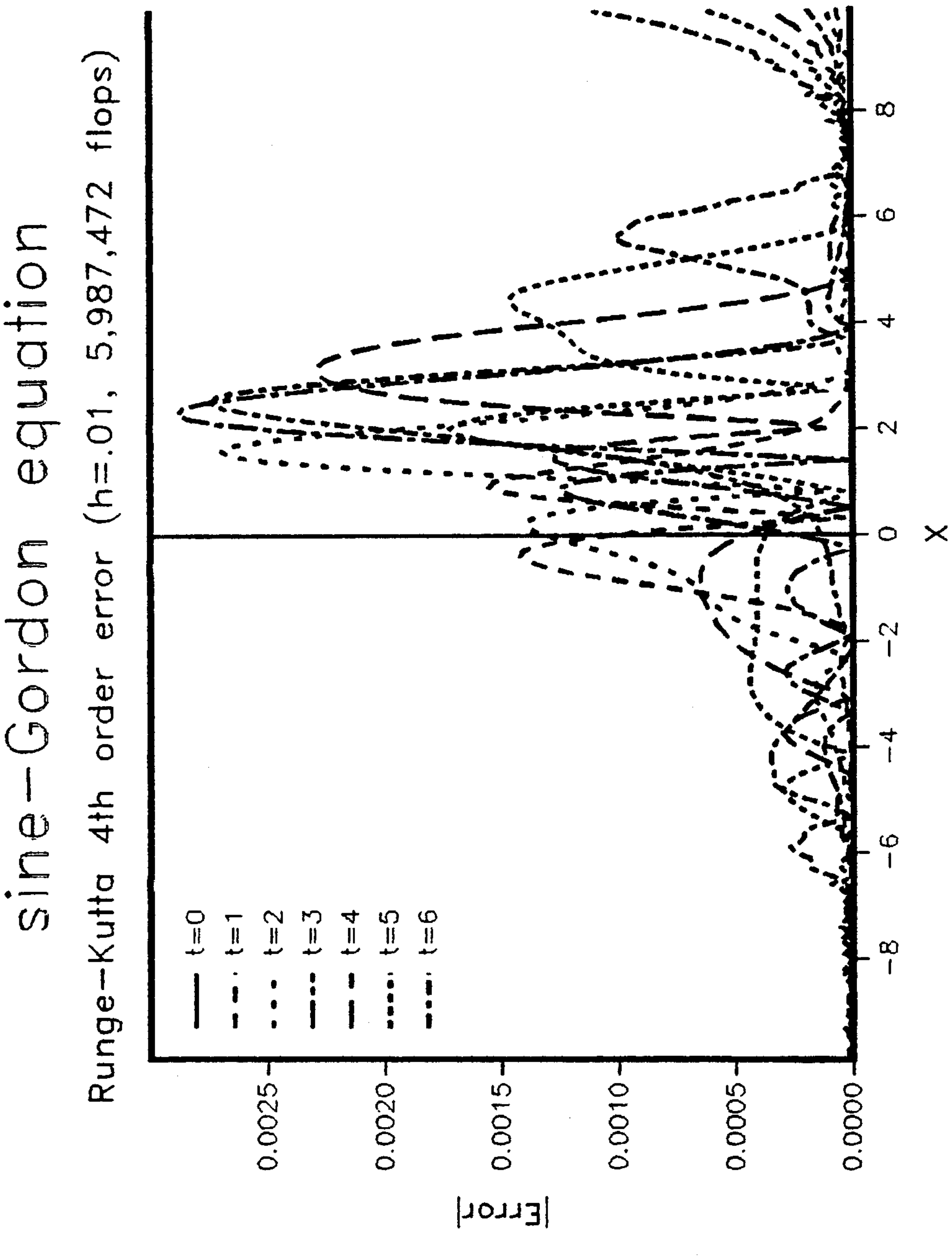


Figure 3.39 sine-Gordon Equation (Runge-Kutta 4th Order Error

CHAPTER 4 - CONCLUSIONS

As seen in the error plots shown in Chapter 3, the Godunov integration method offers a clear advantage in terms of the largest step size allowed for stability, the number of floating point operations required for a solution and in the magnitude of the discretization error over the Adams-Bashforth 3rd order and the Runge-Kutta 4th order methods when used to solve the simple system

$$\ddot{\underline{x}}(t) = -\omega^2 \underline{x}(t) \tag{4.1}$$

where ω^2 is derived from the use of a centered difference scheme to discretize $\frac{\partial^2 U}{\partial x^2}$.

The combined Godunov and Adams-Bashforth 3rd order method offers a small advantage over the Adams-Bashforth 3rd method in terms of discretization error and the number of floating point operations required for a solution when used to solve the slightly more complex system

$$\ddot{\underline{x}}(t) = -\omega^2 \underline{x}(t) + C\dot{\underline{x}}(t) \tag{4.2}$$

where ω^2 and C arise from a second order differential equation formulation of the human body model. Due to the different shapes of the stability domains for the two different methods (Figure 2.1), the eigenvalues of the the system in question will determine which integration technique allows the largest step size before instability occurs. That is why, in this case, the Adams-Bashforth 3rd order method went unstable at a slightly larger step size than did the Godunov method.

For the more complicated systems

$$\ddot{\underline{x}}(t) = \underline{f}_1(\underline{x}_1(t), \dot{\underline{x}}_1(t), x_2(t), u(t), t) \tag{4.3a}$$

$$\dot{x}_2(t) = \underline{f}_2(\underline{x}_1(t), x_2(t), t) \tag{4.3b}$$

and $$\ddot{\underline{x}}(t) = \underline{f}(\underline{x}(t), \dot{\underline{x}}(t), \dot{u}(t), t) \tag{4.4}$$

where $\underline{f}_1$, $\underline{f}_2$ and $\underline{f}$ are defined by the electrical systems experimented with in Chapter 3, the Adams-Bashforth 3rd order method was shown to be superior in terms of the largest stepsize allowed for stability, the number of floating point operations required for a solution and in the magnitude of the discretization error. In addition, it was found that the Godunov method doesn't have any stability domain when used to solve the system

$$\begin{aligned} \ddot{\underline{x}}(t) &= \underline{f}_1(\underline{x}_1(t), \dot{\underline{x}}_1(t), x_2(t), \dot{u}(t), t) \\ \dot{x}_2(t) &= f_2(\underline{x}_1(t), x_2(t), t) \quad . \end{aligned} \tag{4.5}$$

The failure of the Godunov method to converge to the analytic solution in this case can be attributed to the fact that the numerical system has two eigenvalues lying at $z = 1$, or $s = 0$, causing a ramp response. The Adams-Bashforth 3rd order method as well as higher order Adams-Bashforth methods and Adams-Moulton predictor-corrector schemes were used on the first order equations of equation set (4.5), all with the same results, namely a ramp output indicating two eigenvalues at the origin.

Due to the shapes of their respective stability domains (Figure 2.1), the Godunov method will always have an advantage over the Adams-Bashforth 3rd order method in terms of the largest step size allowed for stability when used to solve problems of the type $\ddot{\underline{x}}(t) = -\omega^2 \underline{x}$ which have the eigenvalues of the corresponding system

$$\dot{\underline{x}} = \begin{pmatrix} 0 & I \\ -\omega^2 & 0 \end{pmatrix} \underline{x}$$

lying near the imaginary axis. To check and see if the Godunov method will always offer a floating point operations count advantage over the Adams-Bashforth 3rd order method for problems of this type, a worst case study can be done. This is done by assuming that the ω^2 matrix has no elements whose values are zero. If the vector $\underline{x}$ has n elements, then the evaluations of the 2nd derivative will require $2n^2 - n$ floating point operations. A Godunov solution will then require $4n + 2n^2 - n$ or $2n^2 + 3n$ floating point operations per step. If the n second derivative equations of $\ddot{\underline{x}} = -\omega^2\underline{x}$ are separated into $2n$ 1st derivative equations, the resulting system will be $\dot{\underline{x}} = A\underline{x}$ where

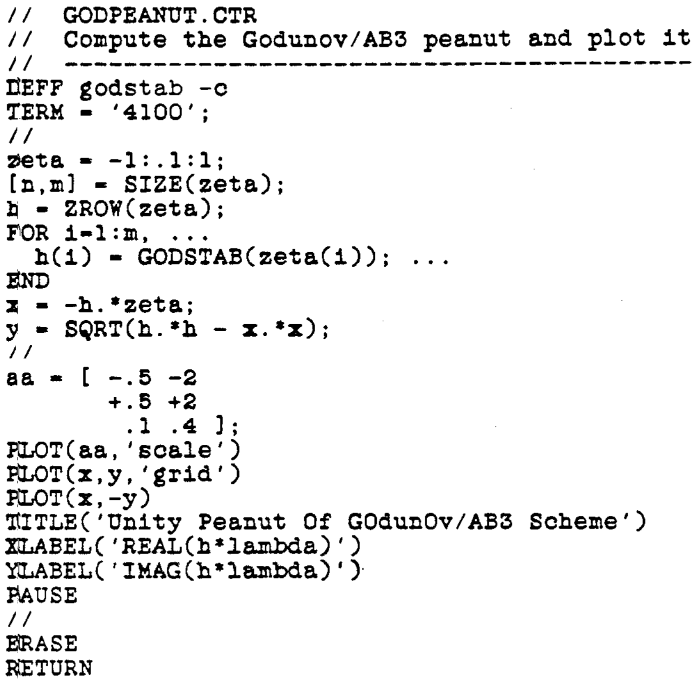
$$A = \begin{pmatrix} 0 & I \\ -\omega^2 & 0 \end{pmatrix}$$

which requires $2n^2 - 1$ floating point operations to evaluate. An Adams-Bashforth 3rd order solution will then require $2n^2 + 14n - 1$ floating point operations per step giving the Godunov method an advantage of $11n - 1$ fewer floating point operations per step.

The bottom line of all this is that the Godunov method may, in certain cases, be a more desirable integration technique than other numerical integration techniques such as the Adams-Bashforth and Runge-Kutta methods. These cases include, but of course are not limited to, systems with eigenvalues lying near the imaginary axis and systems which naturally give rise to 2nd order differential equations such as the mechanical systems described in Chapter 1. When systems require the differentiation of integral terms to be formulated in terms of 2nd order differential equations, such as electrical systems with both capacitors and inductors, however, the Godunov method may not produce a satisfactory solution to the problem.

APPENDIX A - NUMERICAL STABILITY DOMAIN

Appendix A contains a listing of the Control-C code used to generate both the Adams-Bashforth 3rd order and the Godunov–Adams-Bashforth 3rd order hybrid method stability domains found in Figure (2.1).



```
//  GODPEANUT.CTR
//  Compute the Godunov/AB3 peanut and plot it
//  ------------------------------------------
DEFF godstab -c
TERM = '4100';
//
zeta = -1:.1:1;
[n,m] = SIZE(zeta);
h = ZROW(zeta);
FOR i=1:m, ...
  h(i) = GODSTAB(zeta(i)); ...
END
x = -h.*zeta;
y = SQRT(h.*h - x.*x);
//
aa = [ -.5 -2
       +.5 +2
        .1 .4 ];
PLOT(aa,'scale')
PLOT(x,y,'grid')
PLOT(x,-y)
TITLE('Unity Peanut Of GOdunOv/AB3 Scheme')
XLABEL('REAL(h*lambda)')
YLABEL('IMAG(h*lambda)')
PAUSE
//
ERASE
RETURN
```

Figure A.1 GODPEANUT.CTR

```
// [h] = GODSTAB(zeta)
//
//  Compute the eigenvalues of the Godunov/AB3
//  scheme for second order system
//  q(s) = s*s + 2*zeta*omega0*s + omega0*omega0
//  with omega0 = 1
//  Iterate over values of h until the largest
//  absolute eigenvalue is 1.0
//
lmax = 0;
hmin = 0.02;
h = hmin;
WHILE  lmax <= (1+10000*eps), ...
  A = [(2-h*h) -2*h*h*zeta;-23*h/12 (1-23*h*zeta/6)];...
  B = [ -1 0 ; 4*h/3 8*h*zeta/3 ]; ...
  C = [ 0 0 ; -5*h/12 -5*h*zeta/6 ]; ...
  I = EYE(2); ...
  Z = ZROW(2); ...
  F = [ Z I Z ; Z Z I ; C B A ]; ...
  l = EIG(F); ...
  lmax = MAX(ABS(l)); ...
  err = ABS(lmax - 1); ...
  h = h + hmin; ...
END
//
err = 1;
hmax = h;
hmin = h - 2*hmin;
WHILE err > 0.00001, ...
  h = 0.5*(hmin + hmax); ...
  A = [(2-h*h) -2*h*h*zeta;-23*h/12 (1-23*h*zeta/6)];...
  B = [ -1 0 ; 4*h/3 8*h*zeta/3 ]; ...
  C = [ 0 0 ; -5*h/12 -5*h*zeta/6 ]; ...
  I = EYE(2); ...
  Z = ZROW(2); ...
  F = [ Z I Z ; Z Z I ; C B A ]; ...
  l = EIG(F); ...
  lmax = MAX(ABS(l)); ...
  err = ABS(lmax - 1); ...
  hdiff = 0.5*(hmax - hmin); ...
  IF lmax > 1.0, ...
    hmax = hmax - hdiff; ...
   ELSE ...
    hmin = hmin + hdiff; ...
  END, ...
END
//
RETURN
```

Figure A.2 GODSTAB.CTR

```
//  AB3PEANUT.CTR
//  Compute the AB3 peanut and plot it
//  ---------------------------------
DEFF ab3stab -c
TERM = '4100';
//
zeta = -1:.1:1;
[n,m] = SIZE(zeta);
h = ZROW(zeta);
FOR i=1:m, ...
  h(i) = AB3STAB(zeta(i)); ...
END
x = -h.*zeta;
y = SQRT(h.*h - x.*x);
//
aa = [ -1 -1
       +1 +1
       .2 .2 ];
PLOT(aa,'scale')
PLOT(x,y,'grid')
PLOT(x,-y)
TITLE('Unity Peanut Of Adams/BashfOrth 3rd Order Scheme')
XLABEL('REAL(h*lambda)')
YLABEL('IMAG(h*lambda)')
PAUSE
//
ERASE
RETURN
```

Figure A.3 AB3PEANUT.CTR

```
// [h] = AB3STAB(zeta)
//
//  Compute the eigenvalues of the Adams/Bashforth 3rd order
//  scheme q(s) = s*s + 2*zeta*omega0*s + omega0*omega0
//  with omega0 = 1
//  Iterate over values of h until the largest absolute
//  eigenvalue is 1.0
//
lmax = 0;
hmin = 0.02;
h = hmin;
WHILE  lmax <= (1+10000*eps), ...
  A = [ 1 23*h/12 ; -23*h/12 (1-23*h*zeta/6) ]; ...
  B = [ 0 -4*h/3 ; 4*h/3 8*h*zeta/3 ]; ...
  C = [ 0 5*h/12 ; -5*h/12 -5*h*zeta/6 ]; ...
  I = EYE(2); ...
  Z = ZROW(2); ...
  F = [ Z I Z ; Z Z I ; C B A ]; ...
  l = EIG(F); ...
  lmax = MAX(ABS(l)); ...
  err = ABS(lmax - 1); ...
  h = h + hmin; ...
END
//
err = 1;
hmax = h;
hmin = h - 2*hmin;
WHILE err > 0.00001, ...
  h = 0.5*(hmin + hmax); ...
  A = [ 1 23*h/12 ; -23*h/12 (1-23*h*zeta/6) ]; ...
  B = [ 0 -4*h/3 ; 4*h/3 8*h*zeta/3 ]; ...
  C = [ 0 5*h/12 ; -5*h/12 -5*h*zeta/6 ]; ...
  I = EYE(2); ...
  Z = ZROW(2); ...
  F = [ Z I Z ; Z Z I ; C B A ]; ...
  l = EIG(F); ...
  lmax = MAX(ABS(l)); ...
  err = ABS(lmax - 1); ...
  hdiff = 0.5*(hmax - hmin); ...
  IF lmax > 1.0, ...
    hmax = hmax - hdiff; ...
   ELSE ...
    hmin = hmin + hdiff; ...
  END, ...
END
//
RETURN
```

Figure A.4 AB3STAB.CTR

APPENDIX B - COMPUTER CODE USED FOR NUMERICAL SOLUTIONS

Appendix B contains a complete listing of the computer code used to generate the numerical solutions and the analytic solutions to the examples experimented with in this thesis. The Control-C code, which was first used to determine if there was any merit to the idea of using the Godunov method to solve second order differential equations, is listed on pages 120 through 131. This is followed by a listing of the C code, which was used to obtain solutions in a more timely fashion. Both programs were run on a VAX 11/780.

```
// file name hpdetest.ctr
xc=[]
y1=[]
y2=[]
t=[]
diary ›hpdetest.dia
deff tse -c
deff fe -c
deff ab2 -c
deff ab3 -c
deff godstp -c
deff godu -c
deff f1 -c
deff f2 -c
deff g -c
deff f -c
deff g1 -c
deff adms -c
dx=.1;
aa1=zrow(10);
bb1=(1/(dx*dx))*[-2 1 0 0 0 0 0 0 0 0
   1 -2 1 0 0 0 0 0 0 0
   0 1 -2 1 0 0 0 0 0 0
   0 0 1 -2 1 0 0 0 0 0
   0 0 0 1 -2 1 0 0 0 0
   0 0 0 0 1 -2 1 0 0 0
   0 0 0 0 0 1 -2 1 0 0
   0 0 0 0 0 0 1 -2 1 0
   0 0 0 0 0 0 0 1 -2 1
   0 0 0 0 0 0 0 0 2 -2];
cc1=zrow(10);
dd1=zrow(10,1);
ee1=zrow(10);
ff1=zrow(10);
gg1=zrow(10);
hh1=zrow(10,1);
ii1=zrow(10);
jj1=eye(10);
kk1=zrow(10);
ll1=zrow(10,1);
a21=[-5 4 -1 0 0 0 0 0 0 0
     -2 1  0 0 0 0 0 0 0 0
      1 -2 1 0 0 0 0 0 0 0
      0 1 -2 1 0 0 0 0 0 0
      0 0 1 -2 1 0 0 0 0 0
      0 0 0 1 -2 1 0 0 0 0
      0 0 0 0 1 -2 1 0 0 0
      0 0 0 0 0 1 -2 1 0 0
      0 0 0 0 0 0 1 -2 1 0
      0 0 0 0 0 0 0 1 -2 1
      0 0 0 0 0 0 0 0 2 -2];
a=[zrow(10,11),eye(10);a21/(dx*dx),zrow(11)]
eig(a)
eig(bb1)
pause
b=zrow(21,1)
c=eye(10);
c=[c,zrow(10,11)]
d=zrow(10,1)
pi2=pi/2;
x10=zrow(10,1);
x0=zrow(21,1);
for i=1:10,...
   x10(i,1)=sin(pi2*i*dx);...
```

Figure B.1 HPDETEST.CTR

```
   x0(i,1)=sin(pi2*i*dx);...
end
x10
x0
x1d0=zrow(10,1)
x20=zrow(10,1)
dt=.01;
t=[0:dt:10];
[m,n]=size(t);
u=zrow(t);
chop(0); // initialize floating point operations counter
fl1=flop(2);
// first, the Godunov solution
[y1]=godu(aa1,bb1,cc1,dd1,ee1,ff1,gg1,hh1,ii1,jj1,kk1,ll1,x10,x1d0,x20,u,dt,t);
fl2=flop(2);
fl1=fl2-fl1; // to keep track of number of flops for each method
save dt t fl1 y1 >hpdetestgo.dat
simu('ic',x0)
[yex]=simu(a,b,c,d,u,t);
save dt t fl1 y1 >hpdetestsim.dat
fl2=flop(2);
[yext1]=adms(a,b,c,d,x0,u,dt,t);
fl3=flop(2);
fl2=fl3-fl2;
save dt t fl2 yext1 >hpdetestadms.dat
// calculate analytic solution
for i=1:n,...
   for j=1:10,...
      xc(j,i)=sin(pi2*j*dx)*cos(pi2*t(i));...
   end,...
end
save dt t xc >hpdetestan.dat
diary -off
//quit
//$ exit
```

Figure B.1 HPDETEST.CTR

```
// file name humbod.ctr
deff tse -c
deff fe -c
deff ab2 -c
deff ab3 -c
deff godstp -c
deff godu -c
deff f1 -c
deff f2 -c
deff g -c
deff f -c
deff g1 -c
deff adms -c
//deff kcalc -c
m1=1.2;
k1=.3;
b1=.8;
m2=14;
k2=8;
b2=10;
m3=3.2;
k3=3;
b3=12;
m4=24;
kk = k1 + k2 + k3;
bb = b1 + b2 + b3;
aa1=[(-b1/m1)  (b1/m1)  0   0
   (b1/m2) ((-b1-b2-b3)/m2) (b2/m2) (b3/m2)
      0      (b2/m3)  (-b2/m3)  0
      0      (b3/m4)      0     (-b3/m4)];
bb1=[(-k1/m1)  (k1/m1)  0   0
   (k1/m2) ((-k1-k2-k3)/m2) (k2/m2) (k3/m2)
      0      (k2/m3)  (-k2/m3)  0
      0      (k3/m4)      0     (-k3/m4)];
cc1=zrow(4);
dd1=[0;0;0;(1/m4)];
ee1=zrow(4);
ff1=zrow(4);
gg1=zrow(4);
h1=zrow(4,1);
ii1=zrow(1,4);
jj1=[1 -1 0 0];
kk1=zrow(1,4);
ll1=0;
//at1 =[ 0       0       0       0       1       0       0       0
//       0       0       0       0       0       1       0       0
//       0       0       0       0       0       0       1       0
//       0       0       0       0       0       0       0       1
//     -k1/m1  k1/m1   0       0     -b1/m1  b1/m1   0       0
//       k1/m2 -kk/m2  k2/m2   k3/m2  b1/m2 -bb/m2  b2/m2  b3/m2
//       0       k2/m3 -k2/m3  0       0       b2/m3 -b2/m3  0
//       0       k3/m4  0      -k3/m4  0       b3/m4  0      -b3/m4 ]
a =  [ 0       1       0       0       0       0       0       0
      -k1/m1 -b1/m1  k1/m1   b1/m1   0       0       0       0
       0       0       0       1       0       0       0       0
       k1/m2  b1/m2 -kk/m2  -bb/m2  k2/m2   b2/m2  k3/m2   b3/m2
       0       0       0       0       0       1       0       0
       0       0      k2/m3   b2/m3 -k2/m3 -b2/m3  0       0
       0       0       0       0       0       0       0       1
       0       0      k3/m4   b3/m4  0       0      -k3/m4 -b3/m4 ];
b = [ 0 ; 0 ; 0 ; 0 ; 0 ; 0 ; 0 ; 1/m4];
c = [ 1 0 -1 0 0 0 0 0 ];
d = 0;
x10=[1;6;.5;2];
```

Figure B.2 HUMBOD.CTR

```
x1d0=[0;0;0;0];
x20=[0;0;0;0];
//dt=.01;
t=[0:h:1000*h];
//t=[0:dt:10];
[m,n]=size(t);
u=zrow(t);
for i=1:n,...
   u(1,i)=10*sin(pi*t(i));...
end
flop(0); // initialize floating point operations counter
fl1=flop(2);
// first calculate Godunov solution
[y]=godu(aa1,bb1,cc1,dd1,ee1,ff1,gg1,h1,ii1,j1,kk1,ll1,x10,x1d0,x20,u,dt,t);
fl2=flop(2);
fl1=fl2-fl1;
save t dt y >humbodg.dat
// the next set of commented out lines are used to calculate the Godunov
// solution using the equation x(n+1)=A*x(n)
//w2=(-1)*bb1;
//be=(-1)*aa1;
//h=.01;
//e0=zrow(4);
//e1=eye(4);
//e2=2*e1-(h**2)*w2;
//e3= -(h**2)*be;
//e4=(4/3)*h*w2;
//e5=(4/3)*h*be;
//e6= -(5/12)*h*w2;
//e7= -(5/12)*h*be;
//e8= -(23/12)*h*w2;
//e9=e1-(23/12)*h*be;
//ae=[e0 e0 e0 e0 e1 e0
//e1 e0 e0 e0 e0 e0
//e0 e0 e0 e0 e0 e1
//e0 e0 e1 e0 e0 e0
//-e1 e0 e0 e0 e2 e3
//e4 e6 e5 e7 e8 e9];
//r=eig(ae)
//return
//y0=[1;6;.5;2];
//v0=[1;6;.5;2];
//w0=[0;0;0;0];
//u0=[0;0;0;0];
//x0=[1;6;.5;2];
//z0=[0;0;0;0];
//va0=[y0;v0;w0;u0;x0;z0];
//vn=va0;
//yd(1)=vn(17)-vn(18);
//for i=1:1000,...
//  vn=ae*vn;...
 //  yd(i+1)=vn(17)-vn(18);...
//end
//save t dt yd >humbode.dat
//return
//x0=[1;0;6;0;.5;0;2;0];
//simu('ic',x0)
//[yex,xex]=simu(a,b,c,d,u,t);
//return;
fl2=flop(2);
[yext]=adms(a,b,c,d,x0,u,dt,t);
fl3=flop(2);
fl2=fl3-fl2;
//
```

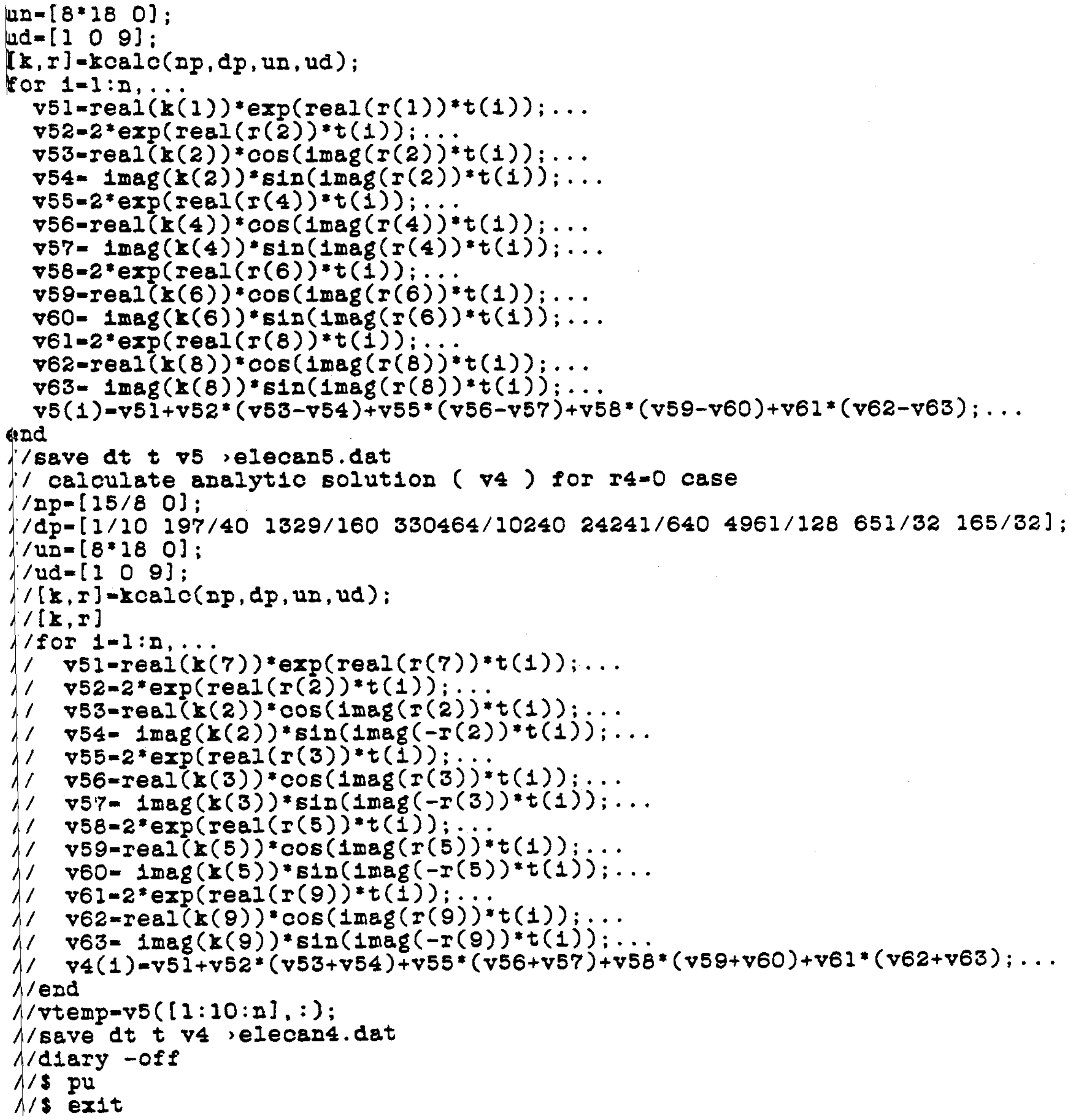
Figure B.2 HUMBOD.CTR

```
// file name elec.ctr
deff tse -c
deff fe -c
deff ab2 -c
deff ab3 -c
deff godstp -c
deff godu -c
deff f1 -c
deff f2 -c
deff g -c
deff f -c
deff g1 -c
deff adms -c
deff kcalc -c
r1=1;
r2=2;
r3=1/4;
r4=1/2;
//r4=0;
c1=1;
c2=2;
c3=1/2;
c4=1/10;
l1=8/5;
l2=4/3;
l3=1/2;
//
aa1=[(-1/(r1*c1))           0            0            0
          0         (-1/(r2*c2))       0            0
          0               0       (-1/(r3*c3))  (1/(r3*c3))
          0               0        (1/(r3*c4))  (-1/(r3*c4))];
//
bb1=[(-1/(l1*c1))    (1/(l1*c1))       0            0
      (1/(l1*c2))  -((l1+l2)/(l1*c2*l2)) (1/(l2*c2)) 0
         0 (1/(c3*l2)) (-1/(c3*l2)) 0
         0 0 0 (-1/(l3*c4))];
//
cc1=[0;0;0;(1/(l3*c4))];
//
dd1=[(1/c1);0;0;0];
//
ee1=zrow(1,4);
//
ff1=[0,0,0,(r4/l3)];
//
gg1=[(-r4/l3)];
//
h1=0;
//
ii1=zrow(1,4);
//
j1=zrow(1,4);
//j1=[0 0 0 1];
//
kk1=[1];
//kk1=[0];
//
ll1=0;
//
a=[(-1/(r1*c1)) (-1/c1) 0 0 0 0 0
(1/l1) 0 (-1/l1) 0 0 0 0
0 (1/c2) (-1/(c2*r2)) (-1/c2) 0 0 0
0 0 (1/l2) 0 (-1/l2) 0 0
0 0 0 (1/c3) (-1/(c3*r3)) (1/(c3*r3)) 0
```

Figure B.3 ELEC.CTR

```
0 0 0 0 (1/(c4*r3)) (-1/(c4*r3)) (-1/c4)
0 0 0 0 0 (1/l3) (-r4/l3)];
//
b=[1/c1;0;0;0;0;0;0];
//
c=[0,0,0,0,0,0,r4];
//c=[0,0,0,0,0,1,0];
//
d=[0];
x10=[10;0;0;0];
//x1d0=[144;0;0;0];
x1d0=[0;0;0;0];
x20=[0];
dt=h;
t=[0:dt:10];
[m,n]=size(t);
u=zrow(t);
x10=10;
x1d0=0;
x20=0;
bb1= -1;
aa1= -1;
cc1=1;
ee1=0;
ff1= .5;
dd1=1;
gg1= -.5;
h1=0;
ii1=0;
j1=0;
kk1=1;
ll1=0;
for i=1:n,...
   u(1,i) = -432*sin(3*t(i));...
end
//   u(1,i)=10*sin(3*pi*t(i));...
//   u(1,i) =30*pi*cos(3*pi*t(i));...
//   u(1,i) = 144*cos(3*t(i));...
chop(0); // initialize floating point operation counter
fl1=flop(2);
// first, calculate Godunov solution
[yg]=godu(aa1,bb1,cc1,dd1,ee1,ff1,gg1,h1,ii1,j1,kk1,ll1,x10,x1d0,x20,u,dt,t);
fl2=flop(2);
fl1=fl2-fl1;
save dt t yg >elecgo.dat
//a=[-1,-1;1,-.5];
//b=[1;0];
//c=[0 .5];
//d=0;
//x0=[10;0];
//simu('ic',x0)
//[yex]=simu(a,b,c,d,u,t); // CTRL-C solution
//fl2=flop(2);
//save dt t yex >elecsim.dat
//[yext1]=adms(a,b,c,d,x0,u,dt,t); // do homemade AB3 solution since
//                                   CTRL-C doesn't count flops
//fl3=flop(2);
//fl2=fl3-fl2;
//save dt t yext1 fl3 >elecadms.dat
//
// calculate analytic solution ( v5 ) for r4=1/2 case
//
np=[15/4];
dp=[1/5 201/20 2117/80 2469/32 33055/320 64652/640 3669/64 255/16];
```

Figure B.3 ELEC.CTR



```
un=[8*18 0];
ud=[1 0 9];
[k,r]=kcalc(np,dp,un,ud);
for i=1:n,...
  v51=real(k(1))*exp(real(r(1))*t(i));...
  v52=2*exp(real(r(2))*t(i));...
  v53=real(k(2))*cos(imag(r(2))*t(i));...
  v54= imag(k(2))*sin(imag(r(2))*t(i));...
  v55=2*exp(real(r(4))*t(i));...
  v56=real(k(4))*cos(imag(r(4))*t(i));...
  v57= imag(k(4))*sin(imag(r(4))*t(i));...
  v58=2*exp(real(r(6))*t(i));...
  v59=real(k(6))*cos(imag(r(6))*t(i));...
  v60= imag(k(6))*sin(imag(r(6))*t(i));...
  v61=2*exp(real(r(8))*t(i));...
  v62=real(k(8))*cos(imag(r(8))*t(i));...
  v63= imag(k(8))*sin(imag(r(8))*t(i));...
  v5(i)=v51+v52*(v53-v54)+v55*(v56-v57)+v58*(v59-v60)+v61*(v62-v63);...
end
//save dt t v5 >elecan5.dat
// calculate analytic solution ( v4 ) for r4=0 case
//np=[15/8 0];
//dp=[1/10 197/40 1329/160 330464/10240 24241/640 4961/128 651/32 165/32];
//un=[8*18 0];
//ud=[1 0 9];
//[k,r]=kcalc(np,dp,un,ud);
//[k,r]
//for i=1:n,...
//  v51=real(k(7))*exp(real(r(7))*t(i));...
//  v52=2*exp(real(r(2))*t(i));...
//  v53=real(k(2))*cos(imag(r(2))*t(i));...
//  v54= imag(k(2))*sin(imag(-r(2))*t(i));...
//  v55=2*exp(real(r(3))*t(i));...
//  v56=real(k(3))*cos(imag(r(3))*t(i));...
//  v57= imag(k(3))*sin(imag(-r(3))*t(i));...
//  v58=2*exp(real(r(5))*t(i));...
//  v59=real(k(5))*cos(imag(r(5))*t(i));...
//  v60= imag(k(5))*sin(imag(-r(5))*t(i));...
//  v61=2*exp(real(r(9))*t(i));...
//  v62=real(k(9))*cos(imag(r(9))*t(i));...
//  v63= imag(k(9))*sin(imag(-r(9))*t(i));...
//  v4(i)=v51+v52*(v53+v54)+v55*(v56+v57)+v58*(v59+v60)+v61*(v62+v63);...
//end
//vtemp=v5([1:10:n],:);
//save dt t v4 >elecan4.dat
//diary -off
//$ pu
//$ exit
```

Figure B.3 ELEC.CTR

```
// file name humbodan.ctr
// to calculate denominator polynomial
deff kcalc -c
m1=1.2;
k1=.3;
b1=.8;
m2=14;
k2=8;
b2=10;
m3=3.2;
k3=3;
b3=12;
m4=24;
kk = k1 + k2 + k3;
bb = b1 + b2 + b3;
a =  [ 0      1      0      0      0      0      0      0
      -k1/m1 -b1/m1  k1/m1  b1/m1  0      0      0      0
       0      0      0      1      0      0      0      0
       k1/m2  b1/m2 -kk/m2 -bb/m2  k2/m2  b2/m2  k3/m2  b3/m2
       0      0      0      0      0      1      0      0
       0      0      k2/m3  b2/m3 -k2/m3 -b2/m3  0      0
       0      0      0      0      0      0      0      1
       0      0      k3/m4  b3/m4  0      0     -k3/m4 -b3/m4 ];
p1=[1 b1/m1 k1/m1];
p2=[1 bb/m2 kk/m2];
p3=[1 b2/m3 k2/m3];
p4=[1 b3/m4 k3/m4];
p5=[b2/m2 k2/m2];
p6=[-b2/m3 -k2/m3];
p7=[-b3/m2 -k3/m2];
p8=[b3/m4 k3/m4];
p9=[b1/m1 k1/m1];
p10=[-b1/m2 -k1/m2];
p11=conv(p3,p4);
p12=conv(p2,p11);
p13=conv(p6,p4);
p14=conv(p5,p13);
p15=conv(p8,p3);
p16=conv(p7,p15);
p17=p12+[0,0,p14]+[0,0,p16];
p18=conv(p1,p17);
p19=conv(p10,p11);
p20=conv(p9,p19);
p=p18+[0,0,p20];
p21=conv(p3,(-p7));
p22=(conv(p21,p9))/m4;
p23=conv(p1,p3);
p24=(conv(p23,(-p7)))/m4;
//save p p22 p24 >humbodp.dat
np=p22;
dp=p;
//un=[10*pi];
un=[1];
//ud=[1 0 (pi*pi)];
ud=[1];
[k1,r]=kcalc(np,dp,un,ud);
np=p24;
[k2,r]=kcalc(np,dp,un,ud);
save k1 k2 r >humbodp.dat
return
```

Figure B.4 HUMBODAN.CTR

```
// calculate analytic solution ( x1 and x2 ), then output is x1-x2
//
load <humbodp
for i=1:n,...
  x11=real(k1(3))*exp(real(r(3))*t(i));...
  x21=real(k2(3))*exp(real(r(3))*t(i));...
  x12=real(k1(8))*exp(real(r(8))*t(i));...
  x22=real(k2(8))*exp(real(r(8))*t(i));...
  x13=2*exp(real(r(1))*t(i));...
  x23=x13;...
  x14=real(k1(1))*cos(imag(r(1))*t(i));...
  x24=real(k2(1))*cos(imag(r(1))*t(i));...
  x15= -imag(k1(1))*sin(imag(r(1))*t(i));...
  x25= -imag(k2(1))*sin(imag(r(1))*t(i));...
  x16=2*exp(real(r(4))*t(i));...
  x26=x16;...
  x17=real(k1(4))*cos(imag(r(4))*t(i));...
  x27=real(k2(4))*cos(imag(r(4))*t(i));...
  x18= -imag(k1(4))*sin(imag(r(4))*t(i));...
  x28= -imag(k2(4))*sin(imag(r(4))*t(i));...
  x19=2*exp(real(r(6))*t(i));...
  x29=x19;...
  x110=real(k1(6))*cos(imag(r(6))*t(i));...
  x210=real(k2(6))*cos(imag(r(6))*t(i));...
  x111= -imag(k1(6))*sin(imag(r(6))*t(i));...
  x211= -imag(k2(6))*sin(imag(r(6))*t(i));...
  x112=2*exp(real(r(9))*t(i));...
  x212=x112;...
  x113=real(k1(9))*cos(imag(r(9))*t(i));...
  x213=real(k2(9))*cos(imag(r(9))*t(i));...
  x114= -imag(k1(9))*sin(imag(r(9))*t(i));...
  x214= -imag(k2(9))*sin(imag(r(9))*t(i));...
x1(i)=x11+x12+x13*(x14+x15)+x16*(x17+x18)+x19*(x110+x111)+x112*(x113+x114);...
x2(i)=x21+x22+x23*(x24+x25)+x26*(x27+x28)+x29*(x210+x211)+x212*(x213+x214);...
end
ya=x1-x2;
save dt t ya >humbodan.dat
save t yex >humbodsim.dat
save t dt y yd >humboddif.dat
save dt t yext f12 >humbodadms.dat
save dt t y f11 >humbodgo.dat
diary -off
return
//quit
//$ exit
```

Figure B.4 HUMBODAN.CTR

```
//[k,r]=kcalc(np,dp,un,ud)
// does partial fraction decomposition on transfer function with
// numerator polynomial (np), denominator polynomial (dp),
// and input of u numerator (un), and u denominator (ud).
// only works when convolution of dp and ud has distinct roots.
//dp=conv(dp,ud);
np=conv(np,un);
dtem=dp(1);
dp=dp/dtem;
np=np/dtem;
//r=eig(a);
r=roots(dp);
r1=roots(ud);
r=([r',r1'])';
[rw1,cl1]=size(np);
[rw2,cl2]=size(r);
for i=1:rw2,...
  ni=0;...
  for j1=1:cl1,...
    if abs(np(j1))>eps,...
      ni=ni+np(j1)*((r(i))**(cl1-j1));...
    end,...
  end,...
  di=1;...
  for j1=1:rw2;...
    if abs(r(i)-r(j1))>eps,...
      di=di*(r(i)-r(j1));...
    end,...
  end,...
  k(i)=ni/di;...
end
```

Figure B.5 KCALC.CTR

```
%[y]=adms(a,b,c,d,x0,u,dt,t)
%
[m,n]=size(t);
%
i=1;
xo=x0;
xdvo=f(a,b,xo,u(:,i))
yt=g1(c,d,xo,u(:,i));
y(:,i)=yt;
%
i=2;
x=FE(xo,xdvo,dt)
xdo=f(a,b,x,u(:,i))
yt=g1(c,d,x,u(:,i));
y(:,i)=yt;
xo=x;
%
i=3;
x=AB2(xo,xdo,xdvo,dt)
xd=f(a,b,x,u(:,i))
yt=g1(c,d,x,u(:,i));
y(:,i)=yt;
xo=x;
%
for i=4:n,...
   x=AB3(xo,xd,xdo,xdvo,dt),...
   xdvo=xdo;...
   xdo=xd;...
   xd=f(a,b,x,u(:,i)),...
   yt=g1(c,d,x,u(:,i));...
   y(:,i)=yt;...
   xo=x;...
end
```

Figure B.6 ADMS.CTR

```
//[xn]=TSE(x,xd,xdd,dt)
xn=x+dt*xd+((dt*dt)/2)*xdd;

//[xn]=FE(x,xd,dt)
xn=x+dt*xd;

✓//[xn]=AB2(x,xd,xdo,dt)
xn=x+(dt/2)*(3*xd-xdo);

//[xn]=AB3(x,xd,xdo,xdvo,dt)
xn=x+(dt/12)*(23*xd-16*xdo+5*xdvo);

//[xn]=GODSTP(x,xo,xdd,dt)
xn=2*x-xo+(dt*dt*xdd);

//[xdd]=f1(aa,bb,cc,dd,x1,x1d,x2,u,t)
xdd=aa*x1d+bb*x1+cc*x2+dd*u(t);

✓//[x2d]=f2(ee,ff,gg,hh,x1,x1d,x2,u,t)
x2d=ee*x1d+ff*x1+gg*x2+hh*u(t);

✓//[y]=g(ii,jj,kk,ll,x1,x1d,x2,u,t)
y=ii*x1d+jj*x1+kk*x2+ll*u(t);

✓//[xd]=f(a,b,x,u)
xd=a*x+b*u;

✓//[y]=g1(c,d,x,u)
y=c*x+d*u;
```

Figure B.7 FUNCTION EVALS AND NUMERICAL STEPS

```
/* godunov.c */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include time
#define cmax 2501
#define rmax 401
#define umax 10
#define ymax 50
#define fmax 30
#define rkmax 4
#define temax 10
#define smax 10
#define blowup 10000.0
#define rkvh 2
#define oor 1e20
#define oorl 2e-1

double k1m1,b1m1,k1m2,b1m2,kkkm2,bbbm2,k2m2,b2m2,k3m2,b3m2,k2m3,b2m3,
       k3m4,b3m4,m4i;
double r4,l2,r1c1,c1i,l1i,c2r2,c2i,l2i,c3i,c3r3,c4r3,c4i,l3i,r4l3,l1c1,
        l1c2,l1l2l1c,l2c2,l2c3,l3c4;
double ls,cs,r1s,r2s,csi,lsi,r1csi,r2ls,lcsi;
double dx,dx2,pi,pi2,tpi,bb,ab;
double sig,omeg;
long N;

#include <disk4:[beamis.math]godflop.c>
#include <disk4:[beamis.math]godvar.c>
#include <disk4:[beamis.math]ivpschemf.c>
#include <disk4:[beamis.math]godanal.c>

main()

{

void (*f[fmax])(),(*fs[fmax])(),(*fin[fmax])(),(*fout[fmax])();
void (*fan[fmax])(),mechvar(),elecvar(),hpdevar(),elecvars();
double Y[rmax][cmax],X0[rmax],T[cmax],h,a,b,tol,X10[rmax],X1D0[rmax];
double XD0[rmax][smax];
double X20[1],YA[rmax][cmax],YEA[rmax][cmax],U[umax],theta;
int i,j,k,kk,ks,ki,ko,l,m,start,type;
char sti[fmax],sto[fmax],*g[fmax],*gs[fmax],fn1[fmax],fn2[fmax];
FILE *out_file1,*out_file2;

pi=4.*atan(1);pi2=pi/2.;tpi=2.*pi;
/* list of available integration schemes and systems: */
g[0]= "hpde";          gs[0]= "godunovf";
g[1]= "mech";          gs[1]= "godunovt";
g[2]= "elec5a";        gs[2]= "euler";
g[3]= "elec5b";        gs[3]= "leapfroge";
g[4]= "elec4";         gs[4]= "leapfrogr";
g[5]= "sinegordona";   gs[5]= "ab3ea";
g[6]= "sinegordonb";   gs[6]= "ab3el";
g[7]= "sinegordonc";   gs[7]= "ab3r";
g[8]= "stabpean";      gs[8]= "rk4fh";
g[9]= "elecsmall1g";   gs[9]= "rk4ah";
g[10]= "elecsmall1a";
g[11]= "elecsmall2g";
g[12]= "elecsmall2a";
g[13]= "elecsmall3g";
g[14]= "elecsmall3a";
g[15]= "elecsmall4g";
g[16]= "elecsmall4a";
```

Figure B.8 GODUNOV.C

```
f[0]= &HPDE1;          fan[0]= &HPDEAN;
f[1]= &HPDE2;          fan[1]= &HPDEAN;
f[2]= &MECH1;          fan[2]= &MECHAN;
f[3]= &MECH2;          fan[3]= &MECHAN;
f[4]= &ELEC51;         fan[4]= &ELECAN5;
f[5]= &ELEC52A;        fan[5]= &ELECAN5;
f[6]= &ELEC52B;        fan[6]= &ELECAN5;
f[7]= &ELEC41;         fan[7]= &ELECAN4;
f[8]= &ELEC42;         fan[8]= &ELECAN4;
f[9]= &SINGORA1;
f[10]= &SINGORA2;
f[11]= &SINGORB1;
f[12]= &SINGORB2;
f[13]= &SINGORC1;      fan[13]= &SINGORAN;
f[14]= &SINGORC2;      fan[14]= &SINGORAN;
f[15]= &STABPEAN1;     fan[15]= &STABPEANAN;
f[16]= &STABPEAN2;     fan[16]= &STABPEANAN;
f[17]= &ELECSMALL11G;  fan[17]= &ELECSMALLAN1;
f[18]= &ELECSMALL11A;  fan[18]= &ELECSMALLAN1;
f[19]= &ELECSMALL12;   fan[19]= &ELECSMALLAN1;
f[20]= &ELECSMALL21G;  fan[20]= &ELECSMALLAN2;
f[21]= &ELECSMALL21A;  fan[21]= &ELECSMALLAN2;
f[22]= &ELECSMALL22;   fan[22]= &ELECSMALLAN2;
f[23]= &ELECSMALL31G;  fan[23]= &ELECSMALLAN3;
f[24]= &ELECSMALL31A;  fan[24]= &ELECSMALLAN3;
f[25]= &ELECSMALL32;   fan[25]= &ELECSMALLAN3;
f[26]= &ELECSMALL41G;  fan[26]= &ELECSMALLAN4;
f[27]= &ELECSMALL41A;  fan[27]= &ELECSMALLAN4;
f[28]= &ELECSMALL42;   fan[28]= &ELECSMALLAN4;
fin[0]= &HPDEIN;       fout[0]= &HPDEOUT1;
fin[1]= &HPDEIN;       fout[1]= &HPDEOUT2;
fin[2]= &MECHIN;       fout[2]= &MECHOUT1;
fin[3]= &MECHIN;       fout[3]= &MECHOUT2;
fin[4]= &ELECIN1;      fout[4]= &ELECOUT51;
fin[5]= &ELECIN2;      fout[5]= &ELECOUT52A;
fin[6]= &ELECIN2;      fout[6]= &ELECOUT52B;
fin[7]= &ELECIN1;      fout[7]= &ELECOUT41;
fin[8]= &ELECIN2;      fout[8]= &ELECOUT42;
fin[9]= &SINGORIN;     fout[9]= &SINGOROUTA1;
fin[10]= &SINGORIN;    fout[10]= &SINGOROUTA2;
fin[11]= &SINGORIN;    fout[11]= &SINGOROUTB1;
fin[12]= &SINGORIN;    fout[12]= &SINGOROUTB2;
fin[13]= &SINGORIN;    fout[13]= &SINGOROUTC1;
fin[14]= &SINGORIN;    fout[14]= &SINGOROUTC2;
fin[15]= &STABPEANIN;  fout[15]= &STABPEANOUT1;
fin[16]= &STABPEANIN;  fout[16]= &STABPEANOUT2;
fin[17]= &ELECINSM1;   fout[17]= &ELECOUTSM11G;
fin[18]= &ELECINSM1;   fout[18]= &ELECOUTSM11A;
fin[19]= &ELECINSM1;   fout[19]= &ELECOUTSM12;
fin[20]= &ELECINSM1;   fout[20]= &ELECOUTSM21G;
fin[21]= &ELECINSM1;   fout[21]= &ELECOUTSM21A;
fin[22]= &ELECINSM1;   fout[22]= &ELECOUTSM22;
fin[23]= &ELECINSM32;  fout[23]= &ELECOUTSM31G;
fin[24]= &ELECINSM1;   fout[24]= &ELECOUTSM31A;
fin[25]= &ELECINSM32;  fout[25]= &ELECOUTSM32;
fin[26]= &ELECINSM32;  fout[26]= &ELECOUTSM41G;
fin[27]= &ELECINSM1;   fout[27]= &ELECOUTSM41A;
fin[28]= &ELECINSM32;  fout[28]= &ELECOUTSM42;
fs[0]= &GODUNOV;
fs[1]= &EULER;
fs[2]= &LEAPFROG;
fs[3]= &AB3;
fs[4]= &RK4FH;
fs[5]= &RK4AH;
```

Figure B.8 GODUNOV.C

```
    i=5;
    break;
  }
/* type 0 solves xdd=f(x) */
/* type 1 solves xdd=f(x,xd) */
/* type 2 solves x1dd=f1(x1,x1d,x2) */
/*               x2d =f2(x1,x1d,x2) */
switch (j)
  {
  case 0 :
    hpdevar();
    if (i<1)
      {
      type=0;
      j=1;
      ks=10;
      ko=1;
      for (l=1;l<=ks;l++)
        {
        X10[l]=sin(pi2*(double)l*dx);
        X1D0[l]=0.0;
        }
      break;
      }
    else
      {
      ks=20;
      ko=1;
      for (l=1;l<=20;l++)
        X0[l]=0.0;
      for (l=1;l<=10;l++)
        X0[l]=sin(pi2*(double)l*dx);
      break;
      }
    break;
  case 1 :
    mechvar();
    if (i<1)
      {
      type=1;
      j=3;
      ks=4;
      ko=1;
      for (l=1;l<=4;l++)
        {
        X10[l]=0.0;
        X1D0[l]=0.0;
        }
      break;
      }
    else
      {
      j=2;
      ks=8;
      ko=1;
      for (l=1;l<=8;l++)
        X0[l]=0.0;
      break;
      }
    break;
  case 2 :
    r4=.5;
    elecvar();
    if (i<1)
```

Figure B.8 GODUNOV.C

```
    {
    type=2;
    j=5;
    ks=4;
    ko=1;
    for (l=1;l<=7;l++)
      X0[l]=0.0;
    (*fin[4])(&a,U);
    (*f[4])(X0,XD0,U);
    for (l=1;l<=4;l++)
      X10[l]=0.0;
    X1D0[1]=XD0[1][1];
    X1D0[2]=XD0[3][1];
    X1D0[3]=XD0[5][1];
    X1D0[4]=XD0[6][1];
    X20[1]=0.0;
    break;
    }
  else
    {
    j=4;
    ks=7;
    ko=1;
    for (l=1;l<=7;l++)
      X0[l]=0.0;
    break;
    }
  break;
case 3 :
  r4=.5;
  elecvar();
  if (i<1)
    {
    type=1;
    j=6;
    ks=5;
    ko=1;
    for (l=1;l<=5;l++)
      {
      X10[l]=0.0;
      X1D0[l]=0.0;
      }
    X1D0[1]=144.0;
    X20[1]=0.0;
    break;
    }
  else
    {
    j=4;
    ks=7;
    ko=1;
    for (l=1;l<=7;l++)
      X0[l]=0.0;
    break;
    }
  break;
case 4 :
  r4=0;
  elecvar();
  if (i<1)
    {
    type=1;
    j=8;
    ks=4;
```

Figure B.8 GODUNOV.C

```
fprintf(stderr,"available integration methods and systems are:\n\n");
for (i=0;i<=9;i++)
  fprintf(stderr," %s\n",gs[i]);
fprintf(stderr,"\n");
for (i=0;i<=16;i++)
  fprintf(stderr," %s\n",g[i]);
fprintf(stderr,"\n");
fprintf(stderr,"input integration scheme,system,outfile name,stepsize,\n");
fprintf(stderr,"lower limit and upper limit in that order:\n");
scanf("%s%s%s%f%f%f%*c",sti,sto,fn1,&h,&a,&b);
for (j=0;j<=16;j++)
  if (strcmp(sto,g[j])==0) break;
if (j==8)
  {
  fprintf(stderr,"input theta\n");
  scanf("%f%*c",&theta);
  sig=cos((2.0*pi*theta)/360.0);
  omeg=sin((2.0*pi*theta)/360.0);
  fprintf(stderr,"sigma=%1.15f omega=%1.15f\n",sig,omeg);
  }
outfile1=fopen(fn1,"w");
for (i=0;i<=9;i++)
  if (strcmp(sti,gs[i])==0) break;
if (i==9)
  {
  fprintf(stderr,"input error tolerance for RK4 adaptive stepsize routine\n");
  scanf("%f%*c",&tol);
  }
switch (i)
  {
  case 0 :
    start=0;
    break;
  case 1 :
    start=1;
    i=0;
    break;
  case 2 :
    i=1;
    break;
  case 3 :
    start=0;
    i=2;
    break;
  case 4 :
    start=1;
    i=2;
    break;
  case 5 :
    start=0;
    i=3;
    break;
  case 6 :
    start=1;
    i=3;
    break;
  case 7 :
    start=2;
    i=3;
    break;
  case 8 :
    i=4;
    break;
  case 9 :
```

Figure B.8 GODUNOV.C

```
    ko=1;
    for (l=1;l<=4;l++)
      {
      X10[l]=0.0;
      X1D0[l]=0.0;
      }
    X1D0[1]=144;
    X20[1]=0.0;
    break;
    }
  else
    {
    j=7;
    ks=7;
    ko=1;
    for (l=1;l<=7;l++)
      X0[l]=0.0;
    break;
    }
  break;
case 5 :
  singorvar();
  if (i<1)
    {
    type=0;
    j=10;
    ks=9;
    ko=9;
    for (l=1;l<=ks;l++)
      {
      X10[l]=exp(-50*pow(l*dx-.5,2));
      X1D0[l]=0.0;
      }
    break;
    }
  else
    {
    j=9;
    ks=18;
    ko=9;
    for (l=1;l<=ks;l++)
      X0[l]=0.0;
    for (l=1;l<=9;l++)
      X0[l]=exp(-50*pow(l*dx-.5,2));
    break;
    }
  break;
case 6 :
  singorvar();
  if (i<1)
    {
    type=0;
    j=12;
    ks=11;
    ko=11;
    for (l=1;l<=ks;l++)
      {
      X10[l]=exp(-pow((l-1)*dx-.5,2));
      X1D0[l]=0.0;
      }
    break;
    }
  else
    {
```

Figure B.8 GODUNOV.C

```
    j=11;
    ks=22;
    ko=11;
    for (l=1;l<=ks;l++)
      X0[l]=0.0;
    for (l=1;l<=11;l++)
      X0[l]=exp(-pow((l-1)*dx-.5,2));
    break;
    }
  break;
case 7 :
  fprintf(stderr,"ab ? bb?\n");
  scanf("%f%f%*c",&ab,&bb);
  singorvar();
  if (i<1)
    {
    type=0;
    j=14;
    ks=N-1;
    ko=ks;
    for (l=1;l<=ks;l++)
      {
      X10[l]=4*atan(exp((ab+dx*l)/pow(.75,.5)));;
      X1D0[l]=(-2*exp((ab+dx*l)/pow(.75,.5)))/((1+exp((2*(ab+dx*l))/pow(.75,.5
             )))*pow(.75,.5));
      }
    break;
    }
  else
    {
    j=13;
    ks=2*(N-1);
    ko=N-1;
    for (l=1;l<=N-1;l++)
      X0[l]=4*atan(exp((ab+dx*l)/pow(.75,.5)));;
    for (l=N;l<=2*(N-1);l++)
      X0[l]=(-2*exp((ab+dx*(double)(l-N+1))/pow(.75,.5)))/
          ((1+exp((2*(ab+dx*(double)(l-N+1)))/pow(.75,.5)))*pow(.75,.5));
    break;
    }
  break;
case 8 :
  if (i<1)
    {
    type=1;
    j=16;
    ks=1;
    ko=1;
    X10[1]=0.0;
    X1D0[1]=1.0;
    break;
    }
  else
    {
    j=15;
    ks=2;
    ko=1;
    X0[1]=0.0;
    X0[2]=1.0;
    break;
    }
  break;
case 9 :
  if (i<1)
```

Figure B.8 GODUNOV.C

```
    {
    type=1;
    j=19;
    ks=1;
    ko=1;
    X10[1]=0.0;
    X1D0[1]=0.0;
    break;
    }
  else
    {
    j=17;
    ks=2;
    ko=1;
    X0[1]=0.0;
    X0[2]=0.0;
    break;
    }
  break;
case 10 :
  if (i<1)
    {
    type=1;
    j=19;
    ks=1;
    ko=1;
    X10[1]=0.0;
    X1D0[1]=0.0;
    X20[1]=0.0;
    break;
    }
  else
    {
    j=18;
    ks=3;
    ko=1;
    X0[1]=0.0;
    X0[2]=0.0;
    X0[3]=0.0;
    break;
    }
  break;
case 11 :
  if (i<1)
    {
    type=2;
    j=22;
    ks=1;
    ko=1;
    X10[1]=0.0;
    X1D0[1]=0.0;
    X20[1]=0.0;
    break;
    }
  else
    {
    j=20;
    ks=3;
    ko=1;
    X0[1]=0.0;
    X0[2]=0.0;
    X0[3]=0.0;
    break;
    }
```

Figure B.8 GODUNOV.C

```
  break;
case 12 :
  if (i<1)
    {
    type=2;
    j=22;
    ks=1;
    ko=1;
    X10[1]=0.0;
    X1D0[1]=0.0;
    X20[1]=0.0;
    break;
    }
  else
    {
    j=21;
    ks=3;
    ko=1;
    X0[1]=0.0;
    X0[2]=0.0;
    X0[3]=0.0;
    break;
    }
case 13 :
  if (i<1)
    {
    type=1;
    j=25;
    ks=1;
    ko=1;
    X10[1]=0.0;
    X1D0[1]=0.0;
    break;
    }
  else
    {
    j=23;
    ks=2;
    ko=1;
    X0[1]=0.0;
    X0[2]=0.0;
    break;
    }
  break;
case 14 :
  if (i<1)
    {
    type=1;
    j=25;
    ks=1;
    ko=1;
    X10[1]=0.0;
    X1D0[1]=0.0;
    break;
    }
  else
    {
    j=24;
    ks=2;
    ko=1;
    X0[1]=0.0;
    X0[2]=0.0;
    break;
    }
```

Figure B.8 GODUNOV.C

```
  case 15 :
    if (i<1)
      {
      type=2;
      j=28;
      ks=1;
      ko=1;
      X10[1]=0.0;
      X1D0[1]=0.0;
      X20[1]=0.0;
      break;
      }
    else
      {
      j=26;
      ks=3;
      ko=1;
      X0[1]=0.0;
      X0[2]=0.0;
      X0[3]=0.0;
      break;
      }
    break;
  case 16 :
    if (i<1)
      {
      type=2;
      j=28;
      ks=1;
      ko=1;
      X10[1]=0.0;
      X1D0[1]=0.0;
      X20[1]=0.0;
      break;
      }
    else
      {
      j=27;
      ks=2;
      ko=1;
      X0[1]=0.0;
      X0[2]=0.0;
      break;
      }
    break;
  }
if (i>0)
  {
  fprintf(stderr,"just before numerical solution\n");
  (*fs[i])((*f[j]),(*fin[j]),(*fout[j]),a,b,h,ks,ko,tol,Y,T,X0,start);
  fprintf(stderr,"just after numerical solution\n");
  }
else
  {
  fprintf(stderr,"just before numerical solution\n");
  (*fs[i])((*f[j]),(*fin[j]),(*fout[j]),a,b,h,ks,ko,tol,Y,T,X10,
  X1D0,X20,start,type);
  fprintf(stderr,"just after numerical solution\n");
  }
if ((j<9)||(j>12))
  {
  fprintf(stderr,"just before analytic solution\n");
  (*fan[j])(T,Y,YA,YEA,b);
  fprintf(stderr,"just after analytic solution\n");
```

Figure B.8 GODUNOV.C

```
  }
fprintf(out_file1,"%20d floating point operations\n",flopcnt);
i=ENDT(T,b);
for (kk=1;kk<=20;kk++)
  T[i+kk]=oor;
/* write data out to file */
/* if the system picked was the third sine-Gordon equation */
/* make X the independant axis, otherwise t is the independant axis */
if ((j==13)||(j==14))
  {
  fprintf(stderr,"file name for error data\n");
  scanf("%s%*c",fn2);
  out_file2=fopen(fn2,"w");
  fprintf(out_file2,"%20d floating point operations\n",flopcnt);
  for (i=0;i<=600;i+=100)
    {
    fprintf(out_file1,"%1.5f ",T[i]);
    fprintf(out_file2,"%1.5f ",T[i]);
    }
  fprintf(out_file1,"\n");
  fprintf(out_file2,"\n");
  for (k=1;k<=N-1;k++)
    {
    for (i=0;i<=600;i+=100)
      {
      fprintf(out_file1,"%1.5f ",Y[k][i]);
      fprintf(out_file2,"%1.5f ",YEA[k][i]);
      }
    fprintf(out_file1,"%1.3f\n",ab+dx*k);
    fprintf(out_file2,"%1.3f\n",ab+dx*k);
    }
  }
else
  {
  i=0;
  while (T[i] <= b)
    {
    if (j==9||j==10)
      fprintf(out_file1,"%f %f %f %f\n",Y[1][i],Y[3][i],Y[5][i],T[i]);
    else if (j==11||j==12)
      fprintf(out_file1,"%f %f %f %f\n",Y[2][i],Y[4][i],Y[6][i],T[i]);
    else
      fprintf(out_file1,"%3.6f %3.6f %3.17f %3.3f\n",
                        YA[1][i],Y[1][i],YEA[1][i],T[i]);
    if ((b/h)>10000.0)
      i+=100;
    else if ((b/h)>2000.0)
      i+=5;
    else
      i++;
    }
  }

}
```

Figure B.8 GODUNOV.C

```
/* godflop.c */
/* addition,multiplication,division,sine and cosine defined */
/* for floating point operations counting. */
/* subtraction is to be taken as addition of a negative number */

double fladd(),flmult(),fldiv(),flsin(),flcos();
long flopcnt=0;

/******************************************************************************/

double fladd(double x,double y)

{

flopcnt++;
return x+y;

}

/******************************************************************************/

double flmult(double x,double y)

{

flopcnt++;
return x*y;

}
/******************************************************************************/

double fldiv(double x,double y)

{

flopcnt++;
return x/y;

}

/******************************************************************************/

double flsin(double x)

{

flopcnt++;
return sin(x);

}

/******************************************************************************/

double flcos(double x)

{

flopcnt++;
return cos(x);

}
```

Figure B.9 GODFLOP.C

```
/* godvar.c */
/* assigns values to variables used for mechanical,electrical */
/* and hpde systems. */

/*******************************************************************/

/* mechanical system variables */

#define k1 .3
#define m1 1.2
#define b1 .8
#define k2 8.0
#define m2 14.0
#define b2 10.0
#define k3 3.0
#define m3 3.2
#define b3 12.0
#define m4 24.0

/*******************************************************************/

void mechvar()

{

k1m1=k1/m1;b1m1=b1/m1;k1m2=k1/m2;b1m2=b1/m2;kkkm2=(k1+k2+k3)/m2;
bbbm2=(b1+b2+b3)/m2;k2m2=k2/m2;b2m2=b2/m2;k3m2=k3/m2;b3m2=b3/m2;
k2m3=k2/m3;b2m3=b2/m3;k3m4=k3/m4;b3m4=b3/m4;m4i=1/m4;

}

/*******************************************************************/

/* electrical system variables */

#define r1 1.0
#define c1 1.0
#define l1 1.6  /* 8/5 */
#define r2 2.0
#define c2 2.0
#define c3 .5  /* 1/2 */
#define r3 .25 /* 1/4 */
#define c4 .1  /* 1/10 */
#define l3 .5  /* 1/2 */
/* #define r4 .5 */  /* 1/2 */

/*******************************************************************/

void elecvar()

{

l2=4.0/3.0;r1c1=1/(r1*c1);c1i=1/c1;l1i=1/l1;c2r2=1/(c2*r2);c2i=1/c2;
l2i=1/l2;c3i=1/c3;c3r3=1/(c3*r3);c4r3=1/(c4*r3);c4i=1/c4;l3i=1/l3;
r4l3=r4/l3;l1c1=1/(l1*c1);l1c2=1/(l1*c2);l1l2l1c=(l1+l2)/(l1*l2*c2);
l2c2=1/(l2*c2);l2c3=1/(l2*c3);l3c4=1/(l3*c4);

}

/*******************************************************************/

/* hyperbolic partial differential equation variables */

/*******************************************************************/
```

Figure B.10 GODVAR.C

```
void hpdevar()

{

dx=0.1;
dx2= 1.0/(dx*dx);

}

/************************************************************************/

/* sine-gordon partial differential equation variables */

/************************************************************************/

void singorvar()


dx=0.1;
dx2= 1.0/(dx*dx);
N=(int)((bb-ab)/dx);
```

Figure B.10 GODVAR.C

```
/* ivpschemf.c */
/* collection of ODE's and integration schemes for initial value problems */

/* global function for adaptive stepsize routines */
int ENDT();

/********************************************************************/

void MECH1(X,XD,U) /* set of first order equations for human body model */
double X[][smax],XD[][smax],U[];


XD[1][1] = X[2][1];
XD[2][1] = fladd(fladd(flmult(-klm1,X[1][1]),flmult(-blm1,X[2][1])),
           fladd(flmult(klm1,X[3][1]),flmult(blm1,X[4][1])));
XD[3][1] = X[4][1];
XD[4][1] = fladd(fladd(fladd(flmult(klm2,X[1][1]),flmult(blm2,X[2][1])),
           fladd(flmult(-kkkm2,X[3][1]),flmult(-bbbm2,X[4][1]))),
           fladd(fladd(flmult(k2m2,X[5][1]),flmult(b2m2,X[6][1])),
           fladd(flmult(k3m2,X[7][1]),flmult(b3m2,X[8][1]))));
XD[5][1] = X[6][1];
XD[6][1] = fladd(fladd(flmult(k2m3,X[3][1]),flmult(b2m3,X[4][1])),
           fladd(flmult(-k2m3,X[5][1]),flmult(-b2m3,X[6][1])));
XD[7][1] = X[8][1];
XD[8][1] = fladd(fladd(flmult(k3m4,X[3][1]),flmult(b3m4,X[4][1])),
           fladd(fladd(flmult(-k3m4,X[7][1]),flmult(-b3m4,X[8][1])),
           flmult((m4i),U[1])));


/********************************************************************/

void MECH2(X1,X1D,X2,X1DD,X2D,U) /* set of second order equations (human */
                                 /* body model)                          */
double X1[][smax],X1D[][smax],X2[][smax],X1DD[][smax],X2D[][smax],U[];


X1DD[1][1] = fladd(fladd(flmult(-klm1,X1[1][1]),flmult(klm1,X1[2][1])),
             fladd(flmult(-blm1,X1D[1][1]),flmult(blm1,X1D[2][1])));
X1DD[2][1]=fladd(fladd(fladd(flmult(klm2,X1[1][1]),flmult(-kkkm2,X1[2][1])),
             fladd(flmult(k2m2,X1[3][1]),flmult(k3m2,X1[4][1]))),
             fladd(fladd(flmult(blm2,X1D[1][1]),flmult(-bbbm2,X1D[2][1])),
             fladd(flmult(b2m2,X1D[3][1]),flmult(b3m2,X1D[4][1]))));
X1DD[3][1] = fladd(fladd(flmult(k2m3,X1[2][1]),flmult(-k2m3,X1[3][1])),
             fladd(flmult(b2m3,X1D[2][1]),flmult(-b2m3,X1D[3][1])));
X1DD[4][1] = fladd(fladd(flmult(k3m4,X1[2][1]),flmult(-k3m4,X1[4][1])),
             fladd(fladd(flmult(b3m4,X1D[2][1]),flmult(-b3m4,X1D[4][1])),
             flmult(m4i,U[1])));


/********************************************************************/

void ELEC51(X,XD,U) /* set of first order equations (R4 in circuit) */
double X[][smax],XD[][smax],U[];


XD[1][1] = fladd(fladd(flmult(-rlcl,X[1][1]),flmult(-cli,X[2][1])),
           flmult(cli,U[1]));
XD[2][1] = fladd(flmult(lli,X[1][1]),flmult(-lli,X[3][1]));
```

Figure B.11 IVPSCHEMF.C

```
XD[3][1] = fladd(fladd(flmult(c2i,X[2][1]),flmult(-c2r2,X[3][1])),
           flmult(-c2i,X[4][1]));
XD[4][1] = fladd(flmult(l2i,X[3][1]),flmult(-l2i,X[5][1]));
XD[5][1] = fladd(fladd(flmult(c3i,X[4][1]),flmult(-c3r3,X[5][1])),
           flmult(c3r3,X[6][1]));
XD[6][1] = fladd(fladd(flmult(c4r3,X[5][1]),flmult(-c4r3,X[6][1])),
           flmult(-c4i,X[7][1]));
XD[7][1] = fladd(flmult(l3i,X[6][1]),flmult(-r4l3,X[7][1]));


/****************************************************************/

void ELEC52A(X1,X1D,X2,X1DD,X2D,U) /* set of second order equations */
                                   /* (R4 in circuit)               */
double X1[][smax],X1D[][smax],X2[][smax],X1DD[][smax],X2D[][smax],U[];


X1DD[1][1] = fladd(fladd(flmult(-r1c1,X1D[1][1]),flmult(-l1c1,X1[1][1])),
             fladd(flmult(l1c1,X1[2][1]),flmult(c1i,U[1])));
X1DD[2][1] = fladd(fladd(flmult(-c2r2,X1D[2][1]),flmult(l1c2,X1[1][1])),
             fladd(flmult(-l1l2l1c,X1[2][1]),flmult(l2c2,X1[3][1])));
X1DD[3][1] = fladd(fladd(flmult(-c3r3,X1D[3][1]),flmult(c3r3,X1D[4][1])),
             fladd(flmult(l2c3,X1[2][1]),flmult(-l2c3,X1[3][1])));
X1DD[4][1] = fladd(fladd(flmult(c4r3,X1D[3][1]),flmult(-c4r3,X1D[4][1])),
             fladd(flmult(-l3c4,X1[4][1]),flmult(l3c4,X2[1][1])));
X2D[1][1]  = fladd(flmult(r4l3,X1[4][1]),flmult(-r4l3,X2[1][1]));


/****************************************************************/

void ELEC52B(X1,X1D,X2,X1DD,X2D,U) /* alternate formulation of second  */
                                   /* order equations (R4 in circuit)  */
double X1[][smax],X1D[][smax],X2[][smax],X1DD[][smax],X2D[][smax],U[];


X1DD[1][1] = fladd(fladd(flmult(-r1c1,X1D[1][1]),flmult(-l1c1,X1[1][1])),
             fladd(flmult(l1c1,X1[2][1]),flmult(c1i,U[1])));
X1DD[2][1] = fladd(fladd(flmult(-c2r2,X1D[2][1]),flmult(l1c2,X1[1][1])),
             fladd(flmult(-l1l2l1c,X1[2][1]),flmult(l2c2,X1[3][1])));
X1DD[3][1] = fladd(fladd(flmult(-c3r3,X1D[3][1]),flmult(c3r3,X1D[4][1])),
             fladd(flmult(l2c3,X1[2][1]),flmult(-l2c3,X1[3][1])));
X1DD[4][1] = fladd(fladd(flmult(c4r3,X1D[3][1]),flmult(-c4r3,X1D[4][1])),
             fladd(flmult(-l3c4,X1[4][1]),flmult(l3c4,X2[1][1])));
X1DD[5][1] = fladd(flmult(r4l3,X1D[4][1]),flmult(-r4l3,X1D[5][1]));


/****************************************************************/

void ELEC41(X,XD,U) /* set of first order equations (R4=0) */
double X[][smax],XD[][smax],U[];


XD[1][1] = fladd(fladd(flmult(-r1c1,X[1][1]),flmult(-c1i,X[2][1])),
           flmult(c1i,U[1]));
XD[2][1] = fladd(flmult(l1i,X[1][1]),flmult(-l1i,X[3][1]));
XD[3][1] = fladd(fladd(flmult(c2i,X[2][1]),flmult(-c2r2,X[3][1])),
           flmult(-c2i,X[4][1]));
```

Figure B.11 IVPSCHEMF.C

```
XD[4][1] = fladd(flmult(l2i,X[3][1]),flmult(-l2i,X[5][1]));
XD[5][1] = fladd(fladd(flmult(c3i,X[4][1]),flmult(-c3r3,X[5][1])),
           flmult(c3r3,X[6][1]));
XD[6][1] = fladd(fladd(flmult(c4r3,X[5][1]),flmult(-c4r3,X[6][1])),
           flmult(-c4i,X[7][1]));
XD[7][1] = flmult(l3i,X[6][1]);

}

/********************************************************************/

void ELEC42(X1,X1D,X2,X1DD,X2D,U) /* second order equations (R4=0) */
double X1[][smax],X1D[][smax],X2[][smax],X1DD[][smax],X2D[][smax],U[];

{

X1DD[1][1] = fladd(fladd(flmult(-r1c1,X1D[1][1]),flmult(-l1c1,X1[1][1])),
             fladd(flmult(l1c1,X1[2][1]),flmult(c1i,U[1])));
X1DD[2][1] = fladd(fladd(flmult(-c2r2,X1D[2][1]),flmult(l1c2,X1[1][1])),
             fladd(flmult(-l1l2l1c,X1[2][1]),flmult(l2c2,X1[3][1])));
X1DD[3][1] = fladd(fladd(flmult(-c3r3,X1D[3][1]),flmult(c3r3,X1D[4][1])),
             fladd(flmult(l2c3,X1[2][1]),flmult(-l2c3,X1[3][1])));
X1DD[4][1] = fladd(fladd(flmult(c4r3,X1D[3][1]),flmult(-c4r3,X1D[4][1])),
             fladd(flmult(-l3c4,X1[4][1]),flmult(l3c4,X2[1][1])));

}

/********************************************************************/

void HPDE1(X,XD,U) /* first order equations for wave equation */
double X[][smax],XD[][smax],U[];

{

int i;

for (i=1;i<=10;i++)
  XD[i][1] = X[i+10][1];
XD[11][1] = flmult(dx2,fladd(flmult(-2.,X[1][1]),X[2][1]));
for (i=2;i<=9;i++)
  XD[i+10][1]=flmult(dx2,fladd(X[i-1][1],fladd(flmult(-2.,X[i][1]),X[i+1][1])));
XD[20][1] = flmult(2.,flmult(dx2,fladd(X[9][1],-X[10][1])));

}

/********************************************************************/

void HPDE2(X1,X1D,X2,X1DD,X2D,U) /* second order equations for wave equation */
double X1[][smax],X1D[][smax],X2[][smax],X1DD[][smax],X2D[][smax],U[];

{

int i;

X1DD[1][1]  = flmult(dx2,fladd(flmult(-2.,X1[1][1]),X1[2][1]));
for (i=2;i<=9;i++)
X1DD[i][1]=flmult(dx2,fladd(X1[i-1][1],fladd(flmult(-2.,X1[i][1]),X1[i+1][1])));
X1DD[10][1] = flmult(2.,flmult(dx2,fladd(X1[9][1],-X1[10][1])));

}

/********************************************************************/
```

Figure B.11 IVPSCHEMF.C

```
void SINGORA1(X,XD,U) /* with B.C. U(1,t)=U(0,t)=0; */
                      /* first order equations for sine-Gordon */
double X[][smax],XD[][smax],U[];

{

int i;

for (i=1;i<=9;i++)
  XD[i][1] = X[i+9][1];
XD[10][1]=fladd(flmult(dx2,fladd(X[2][1],flmult(-2.,X[1][1]))),-flsin(X[1][1]));
for (i=2;i<=8;i++)
  XD[i+9][1] =
fladd(flmult(dx2,fladd(X[i+1][1],fladd(flmult(-2.,X[i][1]),X[i-1][1]))),
-flsin(X[i][1]));
XD[18][1]=fladd(flmult(dx2,fladd(flmult(-2.,X[9][1]),X[8][1])),-flsin(X[9][1]));

}

/*******************************************************************/

void SINGORA2(X1,X1D,X2,X1DD,X2D,U) /* with B.C. U(1,t)=U(0,t)=0            */
                                    /* second order equations for sine-Gordon */
double X1[][smax],X1D[][smax],X2[][smax],X1DD[][smax],X2D[][smax],U[];

{

int i;

X1DD[1][1]=
fladd(flmult(dx2,fladd(X1[2][1],flmult(-2.,X1[1][1]))),-flsin(X1[1][1]));
for (i=2;i<=8;i++)
  X1DD[i][1]=
  fladd(flmult(dx2,fladd(X1[i+1][1],fladd(flmult(-2.,X1[i][1]),X1[i-1][1])))
   -flsin(X1[i][1]));
X1DD[9][1]  =
fladd(flmult(dx2,fladd(flmult(-2.,X1[9][1]),X1[8][1])),-flsin(X1[9][1]));

}

/*******************************************************************/

void SINGORB1(X,XD,U)/* w/B.C. U(1+dx,t)=U(0,t),U(-dx,t)=U(1,t) */
                     /* first order equations for sine-Gordon   */
double X[][smax],XD[][smax],U[];

{

int i;

for (i=1;i<=11;i++)
  XD[i][1] = X[i+11][1];
XD[12][1]= fladd(flmult(dx2,fladd(X[2][1],fladd(flmult(-2.,X[1][1]),X[11][1]))),
-flsin(X[1][1]));
for (i=2;i<=10;i++)
  XD[i+11][1]=
fladd(flmult(dx2,fladd(X[i+1][1],fladd(flmult(-2.,X[i][1]),X[i-1][1]))),
-flsin(X[i][1]));
XD[22][1] =
fladd(flmult(dx2,fladd(X[1][1],fladd(flmult(-2.,X[11][1]),X[10][1]))),
-flsin(X[11][1]));

}
```

Figure B.11 IVPSCHEMF.C

```
/******************************************************************/

void SINGORB2(X1,X1D,X2,X1DD,X2D,U)/* w/B.C. U(1+dx,t)=U(0,t),U(-dx,t)=U(1,t) */
                                   /* second order equations for sine-Gordon  */
double X1[][smax],X1D[][smax],X2[][smax],X1DD[][smax],X2D[][smax],U[];

{

int i;

X1DD[1][1]  =
fladd(flmult(dx2,fladd(X1[2][1],fladd(flmult(-2.,X1[1][1]),X1[11][1]))),
-flsin(X1[1][1]));
for (i=2;i<=10;i++)
  X1DD[i][1] =
fladd(flmult(dx2,fladd(X1[i+1][1],fladd(flmult(-2.,X1[i][1]),X1[i-1][1]))),
-flsin(X1[i][1]));
X1DD[11][1]  =
fladd(flmult(dx2,fladd(X1[1][1],fladd(flmult(-2.,X1[11][1]),X1[10][1]))),
-flsin(X1[11][1]));

}

/******************************************************************/

void SINGORC1(X,XD,U)/* w/B.C. U(a,t)=0,U(b,t)=2pi                */
                     /* first order equations for sine-Gordon */
double X[][smax],XD[][smax],U[];

{

int i;

for (i=1;i<=(N-1);i++)
  XD[i][1] = X[i+N-1][1];
XD[N][1] = fladd(flmult(dx2,fladd(X[2][1],flmult(-2.,X[1][1]))),
-flsin(X[1][1]));
for (i=2;i<=(N-2);i++)
  XD[i+N-1][1] =
fladd(flmult(dx2,fladd(X[i+1][1],fladd(flmult(-2.,X[i][1]),X[i-1][1]))),
-flsin(X[i][1]));
XD[2*(N-1)][1] =
fladd(flmult(dx2,fladd(tpi,fladd(flmult(-2.,X[N-1][1]),X[N-2][1]))),
-flsin(X[N-1][1]));

}

/******************************************************************/

void SINGORC2(X1,X1D,X2,X1DD,X2D,U) /* w/B.C. U(a,t)=0,U(b,t)=2pi            */
                                    /* second order equations for sine-Gordon */
double X1[][smax],X1D[][smax],X2[][smax],X1DD[][smax],X2D[][smax],U[];

{

int i;

X1DD[1][1]  =
fladd(flmult(dx2,fladd(X1[2][1],flmult(-2.,X1[1][1]))),-flsin(X1[1][1]));
for (i=2;i<=(N-2);i++)
  X1DD[i][1] =
fladd(flmult(dx2,fladd(X1[i+1][1],fladd(flmult(-2.,X1[i][1]),X1[i-1][1]))),
```

Figure B.11 IVPSCHEMF.C

```
-flsin(X1[i][1]));
X1DD[N-1][1]  =
fladd(flmult(dx2,fladd(tpi,fladd(flmult(-2.,X1[N-1][1]),X1[N-2][1]))),
-flsin(X1[N-1][1]));

}

/***********************************************************************/

void STABPEAN1(X,XD,U) /* stability peanut example to test AB3 */
double X[][smax],XD[][smax],U[];

{

XD[1][1]=X[2][1];
XD[2][1]= fladd(-2.0*flmult(sig,X[2][1]),-flmult((sig*sig+omeg*omeg),X[1][1]));

}

/***********************************************************************/

void STABPEAN2(X1,X1D,X2,X1DD,X2D,U)/*stability peanut example to test Godunov*/
double X1[][smax],X1D[][smax],X2[][smax],X1DD[][smax],X2D[][smax],U[];

{

X1DD[1][1]= fladd(-2.0*flmult(sig,X1D[1][1]),
            -flmult((sig*sig+omeg*omeg),X1[1][1]));

}

/***********************************************************************/

void ELECSMALL11G(X,XD,U) /* SYSTEM1A2 */
double X[][smax],XD[][smax],U[];

{

XD[1][1]=X[2][1];
XD[2][1]= fladd(fladd(flmult(-3.0,X[1][1]),-X[2][1]),U[1]);

}

/***********************************************************************/

void ELECSMALL11A(X,XD,U) /* SYSTEM1A1 */
double X[][smax],XD[][smax],U[];

{

XD[1][1]=fladd(U[1],-X[2][1]);
XD[2][1]=fladd(fladd(X[1][1],-X[2][1]),-X[3][1]);
XD[3][1]=flmult(2.0,X[2][1]);

}

/***********************************************************************/

void ELECSMALL12(X1,X1D,X2,X1DD,X2D,U) /* SYSTEM1G2 */
double X1[][smax],X1D[][smax],X2[][smax],X1DD[][smax],X2D[][smax],U[];

{
```

Figure B.11 IVPSCHEMF.C

```
X1DD[1][1]= fladd(fladd(flmult(-3.0,X1[1][1]),-X1D[1][1]),U[1]);

}

/************************************************************************/

void ELECSMALL21G(X,XD,U) /* SYSTEM2A2 */
double X[][smax],XD[][smax],U[];

{

XD[1][1]=X[2][1];
XD[2][1]=fladd(fladd(fladd(flmult(-3.0,X[1][1]),-X[2][1]),
         flmult(2.0,X[3][1])),U[1]);
XD[3][1]=fladd(flmult(2.0,X[1][1]),flmult(-2.0,X[3][1]));

}

/************************************************************************/

void ELECSMALL21A(X,XD,U) /* SYSTEM2A1 */
double X[][smax],XD[][smax],U[];

{

XD[1][1]=fladd(U[1],-X[2][1]);
XD[2][1]=fladd(fladd(X[1][1],-X[2][1]),-X[3][1]);
XD[3][1]=flmult(2.0,fladd(X[2][1],-X[3][1]));

}

/************************************************************************/

void ELECSMALL22(X1,X1D,X2,X1DD,X2D,U) /* SYSTEM2G2 */
double X1[][smax],X1D[][smax],X2[][smax],X1DD[][smax],X2D[][smax],U[];

{

X1DD[1][1]=fladd(fladd(fladd(flmult(-3.0,X1[1][1]),-X1D[1][1]),
           flmult(2.0,X2[1][1])),U[1]);
X2D[1][1]=fladd(flmult(2.0,X1[1][1]),flmult(-2.0,X2[1][1]));

}

/************************************************************************/

void ELECSMALL31G(X,XD,U) /* SYSTEM3A2 */
double X[][smax],XD[][smax],U[];

{

XD[1][1]=X[2][1];
XD[2][1]=fladd(fladd(-X[2][1],-X[1][1]),U[1]);

}

/************************************************************************/

void ELECSMALL31A(X,XD,U) /* SYSTEM3A1 */
double X[][smax],XD[][smax],U[];

{
```

Figure B.11 IVPSCHEMF.C

```
XD[1][1]=fladd(fladd(U[1],-X[1][1]),-X[2][1]);
XD[2][1]=X[1][1];

}

/**********************************************************************/

void ELECSMALL32(X1,X1D,X2,X1DD,X2D,U) /* SYSTEM3G2 */
double X1[][smax],X1D[][smax],X2[][smax],X1DD[][smax],X2D[][smax],U[];

{

X1DD[1][1]=fladd(fladd(-X1D[1][1],-X1[1][1]),U[1]);

}

/**********************************************************************/

void ELECSMALL41G(X,XD,U) /* SYSTEM4A2 */
double X[][smax],XD[][smax],U[];

{

XD[1][1]=X[2][1];
XD[2][1]=fladd(fladd(fladd(-X[1][1],-X[2][1]),X[3][1]),U[1]);
XD[3][1]=fladd(fldiv(X[1][1],2.0),fldiv(X[3][1],-2.0));

}

/**********************************************************************/

void ELECSMALL41A(X,XD,U) /* SYSTEM4A1 */
double X[][smax],XD[][smax],U[];

{

XD[1][1]=fladd(fladd(U[1],-X[1][1]),-X[2][1]);
XD[2][1]=fladd(X[1][1],fldiv(X[2][1],-2.0));

}

/**********************************************************************/

void ELECSMALL42(X1,X1D,X2,X1DD,X2D,U) /* SYSTEM4G2 */
double X1[][smax],X1D[][smax],X2[][smax],X1DD[][smax],X2D[][smax],U[];

{

X1DD[1][1]=fladd(fladd(fladd(-X1[1][1],-X1D[1][1]),X2[1][1]),U[1]);
X2D[1][1]=fladd(fldiv(X1[1][1],2.0),fldiv(X2[1][1],-2.0));

}

/* systems with IN or OUT in their names provide inputs to the differential*/
/* equations and extract outputs to be saved later                         */
/***************************************************************************/

void MECHIN(t,U)
double *t,U[];

{

U[1] = 10 * sin(pi * *t);
```

Figure B.11 IVPSCHEMF.C

```
}

/****************************************************************************/

void ELECIN1(t,U)
double *t,U[];

{

U[1] = 144. * cos(3. * *t);

}

/****************************************************************************/

void ELECIN2(t,U)
double *t,U[];

{

U[1] = -432. *sin(3. * *t);

}

/****************************************************************************/

void EPDEIN(t,U)
double *t,U[];

{

}

/****************************************************************************/

void SINGORIN(t,U)
double *t,U[];

{

}

/****************************************************************************/

void STABPEANIN(t,U)
double *t,U[];

{

}

/****************************************************************************/

void ELECINSM1(t,U)
double *t,U[];

{

U[1]=10.0*sin(3.0*pi* *t);

}
```

Figure B.11 IVPSCHEMF.C

```
/***************************************************************************/

void ELECINSM32(t,U)
double *t,U[];

{

U[1]=30.0*pi*cos(3.0*pi* *t);

}

/***************************************************************************/

void MECHOUT1(X,U,Y,i)
double X[][smax],Y[][cmax],U[];
int *i;

{

Y[1][*i]=X[1][1]-X[3][1];

}

/***************************************************************************/

void MECHOUT2(X1,X1D,X2,U,Y,i)
double X1[][smax],X1D[][smax],X2[][smax],Y[][cmax],U[];
int *i;

{

Y[1][*i]=X1[1][1]-X1[2][1];

}

/***************************************************************************/

void ELECOUT51(X,U,Y,i)
double X[][smax],Y[][cmax],U[];
int *i;

{

Y[1][*i] = r4*X[7][1];

}

/***************************************************************************/

void ELECOUT52A(X1,X1D,X2,U,Y,i)
double X1[][smax],X1D[][smax],X2[][smax],Y[][cmax],U[];
int *i;

{

Y[1][*i] = X2[1][1];

}

/***************************************************************************/

void ELECOUT52B(X1,X1D,X2,U,Y,i)
double X1[][smax],X1D[][smax],X2[][smax],Y[][cmax],U[];
```

Figure B.11 IVPSCHEMF.C

```
int *i;

{

Y[1][*i] = X1[5][1];

}

/******************************************************************************************/

void ELECOUT41(X,U,Y,i)
double X[][smax],Y[][cmax],U[];
int *i;

{

Y[1][*i] = X[6][1];

}

/******************************************************************************************/

void ELECOUT42(X1,X1D,X2,U,Y,i)
double X1[][smax],X1D[][smax],X2[][smax],Y[][cmax],U[];
int *i;

{

Y[1][*i] = X1[4][1];

}

/**************************************************************************************/

void HPDEOUT1(X,U,Y,i)
double X[][smax],Y[][cmax],U[];
int *i;

{

Y[1][*i]=X[10][1];

}

/**************************************************************************************/

void HPDEOUT2(X1,X1D,X2,U,Y,i)
double X1[][smax],X1D[][smax],X2[][smax],Y[][cmax],U[];
int *i;

{

Y[1][*i]=X1[10][1];

}

/**************************************************************************************/

void SINGOROUTA1(X,U,Y,i)
double X[][smax],Y[][cmax],U[];
int *i;

{
```

Figure B.11 IVPSCHEMF.C

```
for (j=1;j<=N-1;j++)
  Y[j][*i]=X[j][1];

}

/*********************************************************************/

void SINGOROUTC2(X1,X1D,X2,U,Y,i)
double X1[][smax],X1D[][smax],X2[][smax],Y[][cmax],U[];
int *i;

{

int j;

for (j=1;j<=N-1;j++)
  Y[j][*i]=X1[j][1];

}

/*********************************************************************/

void STABPEANOUT1(X,U,Y,i)
double X[][smax],Y[][cmax],U[];
int *i;

{

Y[1][*i]=X[1][1];

}

/*********************************************************************/

void STABPEANOUT2(X1,X1D,X2,U,Y,i)
double X1[][smax],X1D[][smax],X2[][smax],Y[][cmax],U[];
int *i;

{

Y[1][*i]=X1[1][1];

}

/*********************************************************************/

void ELECOUTSM11G(X,U,Y,i)
double X[][smax],Y[][cmax],U[];
int *i;

{

Y[1][*i]=X[1][1];

}

/*********************************************************************/

void ELECOUTSM11A(X,U,Y,i)
double X[][smax],Y[][cmax],U[];
int *i;

{
```

Figure B.11 IVPSCHEMF.C

```
int j;

for (j=1;j<=9;j++)
  Y[j][*i]=X[j][1];

}

/************************************************************************/

void SINGOROUTA2(X1,X1D,X2,U,Y,i)
double X1[][smax],X1D[][smax],X2[][smax],Y[][cmax],U[];
int *i;

{

int j;

for (j=1;j<=9;j++)
  Y[j][*i]=X1[j][1];

}

/************************************************************************/

void SINGOROUTB1(X,U,Y,i)
double X[][smax],Y[][cmax],U[];
int *i;

{

int j;

for (j=1;j<=11;j++)
  Y[j][*i]=X[j][1];

}

/************************************************************************/

void SINGOROUTB2(X1,X1D,X2,U,Y,i)
double X1[][smax],X1D[][smax],X2[][smax],Y[][cmax],U[];
int *i;

{

int j;

for (j=1;j<=11;j++)
  Y[j][*i]=X1[j][1];

}

/************************************************************************/

void SINGOROUTC1(X,U,Y,i)
double X[][smax],Y[][cmax],U[];
int *i;

{

int j;
```

Figure B.11 IVPSCHEMF.C

```
Y[1][*i]=X[2][1];

}

/**********************************************************************/

void ELECOUTSM12(X1,X1D,X2,U,Y,i)
double X1[][smax],X1D[][smax],X2[][smax],Y[][cmax],U[];
int *i;

{

Y[1][*i]=X1[1][1];

}

/**********************************************************************/

void ELECOUTSM21G(X,U,Y,i)
double X[][smax],Y[][cmax],U[];
int *i;

{

Y[1][*i]=X[3][1];

}

/**********************************************************************/

void ELECOUTSM21A(X,U,Y,i)
double X[][smax],Y[][cmax],U[];
int *i;

{

Y[1][*i]=X[3][1];

}

/**********************************************************************/

void ELECOUTSM22(X1,X1D,X2,U,Y,i)
double X1[][smax],X1D[][smax],X2[][smax],Y[][cmax],U[];
int *i;

{

Y[1][*i]=X2[1][1];

}

/**********************************************************************/

void ELECOUTSM31G(X,U,Y,i)
double X[][smax],Y[][cmax],U[];
int *i;

{

Y[1][*i]=X[1][1];
```

Figure B.11 IVPSCHEMF.C

```
}

/*****************************************************************/

void ELECOUTSM31A(X,U,Y,i)
double X[][smax],Y[][cmax],U[];
int *i;

{

Y[1][*i]=X[1][1];

}

/*****************************************************************/

void ELECOUTSM32(X1,X1D,X2,U,Y,i)
double X1[][smax],X1D[][smax],X2[][smax],Y[][cmax],U[];
int *i;

{

Y[1][*i]=X1[1][1];

}

/*****************************************************************/

void ELECOUTSM41G(X,U,Y,i)
double X[][smax],Y[][cmax],U[];
int *i;

{

Y[1][*i]=X[3][1];

}

/*****************************************************************/

void ELECOUTSM41A(X,U,Y,i)
double X[][smax],Y[][cmax],U[];
int *i;

{

Y[1][*i]=X[2][1]/2.0;

}

/*****************************************************************/

void ELECOUTSM42(X1,X1D,X2,U,Y,i)
double X1[][smax],X1D[][smax],X2[][smax],Y[][cmax],U[];
int *i;

{

Y[1][*i]=X2[1][1];

}

/*****************************************************************/
```

Figure B.11 IVPSCHEMF.C

```
void EULER(f,fin,fout,a,b,h,ks,ko,tol,Y,T,X0,start)
void (*f)(),(*fin)(),(*fout)();
double a,b,h,tol,X0[],T[],Y[][cmax];
int ks,ko,start;

{

void EULSTEP(),BLWUP();
double XD[rmax][smax],X[rmax][smax],U[umax];
int i,j,m;

for (i=1;i<=ks;i++)
  X[i][1]=X0[i];
i=0;
T[i]= a;
(*fin)(&T[i],U);
(*fout)(X,U,Y,&i);
while (T[i] <= b)
  {
  for (m=1;m<=ks;m++)
    if (fabs(Y[m][i])>blowup)
      {
      BLWUP(Y,T,i,b,ko,h);
      return;
      }
  (*f)(X,XD,U);
  EULSTEP(&h,&ks,X,XD);
  i++;
  for (j=1;j<=ks;j++)
    X[j][1]=X[j][0];
  T[i] = T[i-1] + h;
  (*fin)(&T[i],U);
  (*fout)(X,U,Y,&i);
  }
T[i+1]=oor;

}

/*********************************************************************/

void LEAPFROG(f,fin,fout,a,b,h,ks,ko,tol,Y,T,X0,start)
void (*f)(),(*fin)(),(*fout)();
double a,b,h,tol,X0[],T[],Y[][cmax];
int ks,ko,start;

{

void EULSTEP(),RK4STEP(),LEAPSTEP(),BLWUP(),SHIFTLEAP();
double XD[rmax][smax],X[rmax][smax],U[umax];
int i,j,m;

for (i=1;i<=ks;i++)
  X[i][1]=X0[i];
i=0;
T[i]= a;
(*fin)(&T[i],U);
(*fout)(X,U,Y,&i);
if (start==0)
  {
  (*f)(X,XD,U);
  EULSTEP(&h,&ks,X,XD);
  }
```

Figure B.11 IVPSCHEMF.C

```
else
  RK4STEP((*f),&h,&ks,X,XD,U);
SHIFTLEAP(&i,X,T,&h,&ks);
(*fin)(&T[i],U);
(*fout)(X,U,Y,&i);
while (T[i] <= b)
  {
  for (m=1;m<=ks;m++)
    if (fabs(Y[m][i])>blowup)
      {
      BLWUP(Y,T,i,b,ko,h);
      return;
      }
  (*f)(X,XD,U);
  LEAPSTEP(&h,&ks,X,XD);
  SHIFTLEAP(&i,X,T,&h,&ks);
  (*fin)(&T[i],U);
  (*fout)(X,U,Y,&i);
  }
T[i+1]=oor;

}

/**********************************************************************/

void AB3(f,fin,fout,a,b,h,ks,ko,tol,Y,T,X0,start) /* Adams-Bashforth 3rd order*/
void (*f)(),(*fin)(),(*fout)();
double a,b,h,tol,X0[],T[],Y[][cmax];
int ks,ko,start;

{

void RK4STEP(),LEAPSTEP(),EULSTEP(),AB2STEP(),AB3STEP(),BLWUP(),SHIFTAB();
double XD[rmax][smax],X[rmax][smax],U[umax];
int i,j,m;

for (i=1;i<=ks;i++)
  X[i][1]=X0[i];
i=0;
T[i]= a;
(*fin)(&T[i],U);
(*fout)(X,U,Y,&i);
if (start<2)
  {
  (*f)(X,XD,U);
  EULSTEP(&h,&ks,X,XD);
  }
else
  RK4STEP((*f),&h,&ks,X,XD,U);
SHIFTAB(&i,XD,X,T,&h,&ks);
(*fin)(&T[i],U);
(*fout)(X,U,Y,&i);
if (start==0)
  {
  (*f)(X,XD,U);
  AB2STEP(&h,&ks,X,XD);
  }
else if (start==1)
  {
  (*f)(X,XD,U);
  LEAPSTEP(&h,&ks,X,XD);
  }
else
```

Figure B.11 IVPSCHEMF.C

```
  RK4STEP((*f),&h,&ks,X,XD,U);
SHIFTAB(&i,XD,X,T,&h,&ks);
(*fin)(&T[i],U);
(*fout)(X,U,Y,&i);
while (T[i] <= b)
  {
  for (m=1;m<=ks;m++)
    if (fabs(Y[m][i])>blowup)
      {
      BLWUP(Y,T,i,b,ko,h);
      return;
      }
  (*f)(X,XD,U);
  AB3STEP(&h,&ks,X,XD);
  SHIFTAB(&i,XD,X,T,&h,&ks);
  (*fin)(&T[i],U);
  (*fout)(X,U,Y,&i);
  }
T[i+1]=oor;

}

/**********************************************************************/

void RK4FH(f,fin,fout,a,b,h,ks,ko,tol,Y,T,X0,start) /* Runge-Kutta 4th       */
void (*f)(),(*fin)(),(*fout)();                     /* order, fixed stepsize */
double a,b,h,tol,X0[],T[],Y[][cmax];
int ks,ko,start;

{

void RK4STEP(),BLWUP();
double X[rmax][smax],XD[rmax][smax],yo[ymax],U[umax];
int i,j,m;

for (i=1;i<=ks;i++)
  X[i][1]=X0[i];
i=0;
T[i]= a;
(*fin)(&T[i],U);
(*fout)(X,U,Y,&i);
while (T[i] <= b)
  {
  for (m=1;m<=ks;m++)
    if (fabs(Y[m][i])>blowup)
      {
      BLWUP(Y,T,i,b,ko,h);
      return;
      }
  RK4STEP((*f),&h,&ks,X,XD,U);
  i++;
  for (j=1;j<=ks;j++)
    X[j][1]=X[j][0];
  T[i]=T[i-1]+ h;
  (*fin)(&T[i],U);
  (*fout)(X,U,Y,&i);
  }
T[i+1]=oor;

}

/**********************************************************************/
```

Figure B.11 IVPSCHEMF.C

```
void RK4AH(f,fin,fout,a,b,h,ks,ko,tol,Y,T,X0,start) /* Runge-Kutta 4th order */
void (*f)(),(*fin)(),(*fout)();                     /* adaptive stepsize     */
double a,b,h,tol,X0[],T[],Y[][cmax];
int ks,ko,start;

{

void RK4STEP();
double X[rmax][smax],U[umax],XD[rmax][smax],XT[rmax][smax],l,tn,xm;
int i,j;

for (i=1;i<=ks;i++)
  X[i][1]=X0[i];
i=0;
T[i]=a;
(*fin)(&T[i],U);
(*fout)(X,U,Y,&i);
while (T[i]<=b)
  {
  /* take two half steps, one full step and compare */
  RK4STEP((*f),&h,&ks,X,XD,U);
  for (j=1;j<=ks;j++)
    XT[j][2]=X[j][0];
  l=h/2;
  RK4STEP((*f),&l,&ks,X,XD,U);
  for (j=1;j<=ks;j++)
    XT[j][1]=X[j][0];
  tn=T[i]+l;
  (*fin)(&tn,U);
  RK4STEP((*f),&l,&ks,XT,XD,U);
  /*xm=pow(fabs(XT[1][0]-XT[1][2]),2);*/
  xm=fabs(XT[1][0]-XT[1][2]);
  for (j=2;j<=ks;j++)
    /*xn=fabs(XT[j][0]-XT[j][2]);*/
    /*if (xn>xm)*/
      /*xm=xn;*/
    /*xm+=pow(fabs(XT[j][0]-XT[j][2]),2);*/
    xm+=fabs(XT[j][0]-XT[j][2]);
  /*xm=pow(xm,.5);*/
  /* check to see if stepsize should be doubled, halved or left alone */
  if ((xm/31)<=(tol*h))
    {
    i++;
    T[i] = T[i-1]+ h;
    for (j=1;j<=ks;j++)
      X[j][1]=(32.0*XT[j][0]-XT[j][2])/31.0;
    (*fin)(&T[i],U);
    (*fout)(X,U,Y,&i);
    if ((32.0/31.0)*xm<(tol*h))
      h*=2.0;
    }
  else
    h/=2.0;
  }
T[i+1]=oor;

}

/**************************************************************************/

void GODUNOV(f,fin,fout,a,b,h,ks,ko,tol,Y,T,X10,X1D0,X20,start,type)
void (*f)(),(*fin)(),(*fout)();
double a,b,h,tol,X10[],X1D0[],X20[],T[],Y[][cmax];
```

Figure B.11 IVPSCHEMF.C

```
int ks,ko,start,type;
{

void TAYLORSTEP(),EULSTEP(),AB2STEP(),AB3STEP(),GODSTEP(),BLWUP(),SHIFTGOD();
double X1[rmax][smax],X1D[rmax][smax],X2[rmax][smax],U[umax],X2D[rmax][smax],
       X1DD[rmax][smax];
int i,j,ktemp=1,m;

i=0;
/* first get initial values for X and it's derivative */
for (j=1;j<=ks;j++)
  {
  X1[j][1]=X10[j];
  X1D[j][1]=X1D0[j];
  }
X2[1][1]=X20[1];
T[i]=a;
/* first calculate an input and the output at current time,then   */
/* calculate derivative and current time and use it and the input */
/* to extrapolate to the next time                                */
(*fin)(&T[i],U);
(*fout)(X1,X1D,X2,U,Y,&i);
(*f)(X1,X1D,X2,X1DD,X2D,U);
/* need to do two startup steps before entering main calculation loop */
if (start==0)
  EULSTEP(&h,&ks,X1,X1D);
else
  TAYLORSTEP(&h,&ks,X1,X1D,X1DD);
if (type>0)
  EULSTEP(&h,&ks,X1D,X1DD);
if (type==2)
  EULSTEP(&h,&ktemp,X2,X2D);
SHIFTGOD(&i,X1,X1D,X2,X1DD,X2D,T,&h,&ks);
(*fin)(&T[i],U);
(*fout)(X1,X1D,X2,U,Y,&i);
(*f)(X1,X1D,X2,X1DD,X2D,U);
GODSTEP(&h,&ks,X1,X1DD);
if (type>0)
  AB2STEP(&h,&ks,X1D,X1DD);
if (type==2)
  AB2STEP(&h,&ktemp,X2,X2D);
SHIFTGOD(&i,X1,X1D,X2,X1DD,X2D,T,&h,&ks);
(*fin)(&T[i],U);
(*fout)(X1,X1D,X2,U,Y,&i);
while (T[i]<=b)
  {
  for (m=1;m<=ks;m++)
    if (fabs(Y[m][i])>blowup)
      {
      BLWUP(Y,T,i,b,ko,h);
      return;
      }
  (*f)(X1,X1D,X2,X1DD,X2D,U);
  GODSTEP(&h,&ks,X1,X1DD);
  if (type>0)
    AB3STEP(&h,&ks,X1D,X1DD);
  if (type==2)
    AB3STEP(&h,&ktemp,X2,X2D);
  SHIFTGOD(&i,X1,X1D,X2,X1DD,X2D,T,&h,&ks);
  (*fin)(&T[i],U);
  (*fout)(X1,X1D,X2,U,Y,&i);
  }
```

Figure B.11 IVPSCHEMF.C

```
T[i+1]=oor;

}

/**********************************************************************/

void SHIFTGOD(i,X1,X1D,X2,X1DD,X2D,T,h,ks) /* to shift values in arrays to */
                                           /* make array correct for next  */
                                           /* time step                    */
double X1[][smax],X1D[][smax],X2[][smax],X1DD[][smax],X2D[][smax],T[],*h;
int *i,*ks;

{

int j;

(*i)++;
for (j=1;j<=*ks;j++)
   {
   X1[j][2]=X1[j][1];
   X1[j][1]=X1[j][0];
   X1D[j][1]=X1D[j][0];
   X2[j][1]=X2[j][0];
   X1DD[j][3]=X1DD[j][2];
   X1DD[j][2]=X1DD[j][1];
   X2D[j][3]=X2D[j][2];
   X2D[j][2]=X2D[j][1];
   }
T[*i] = T[*i-1] + *h;

}

/**********************************************************************/

void SHIFTAB(i,XD,X,T,h,ks) /* shifting for Adams-Bashforth 3rd order steps */
double XD[][smax],X[][smax],T[],*h;
int *i,*ks;

{

int j;

(*i)++;
for (j=1;j<=*ks;j++)
   {
   XD[j][3]=XD[j][2];
   XD[j][2]=XD[j][1];
   X[j][1]=X[j][0];
   }
T[*i]=T[*i-1]+ *h;

}

/**********************************************************************/

void SHIFTLEAP(i,X,T,h,ks) /* shifting for LEAPFROG steps */
double X[][smax],T[],*h;
int *i,*ks;

{

int j;
```

Figure B.11 IVPSCHEMF.C

```
(*i)++;
for (j=1;j<=*ks;j++)
  {
  X[j][2]=X[j][1];
  X[j][1]=X[j][0];
  }
T[*i] = T[*i-1] + *h;

}

/*************************************************************************/

int ENDT(T,b) /* to calculate the size of the array T */
double T[],b;

{

int i=0;

while (T[i]<b)
  i++;
return i;

}

/*************************************************************************/

void BLWUP(Y,T,i,b,k,h) /* to output zeroes in case of a numerical catastrophe*/
double h,b,Y[][cmax],T[];
int i,k;

{

int l;

while (T[i]<=b)
  {
  i++;
  T[i]=T[i-1]+h;
  for (l=1;l<=k;l++)
    Y[l][i]=0.;
  }

}

/*************************************************************************/

void TAYLORSTEP(h,k,X,XD,XDD) /* Taylor Series Expansion step */
double *h,X[][smax],XD[][smax],XDD[][smax];
int *k;

{

int i;

for (i=1;i<=*k;i++)
  X[i][0]=fladd(X[i][1],fladd(flmult(*h,XD[i][1]),
flmult((*h * *h)/2.0,XDD[i][1])));

}

/*************************************************************************/
```

Figure B.11 IVPSCHEMF.C

```
void AB2STEP(h,k,X,XD) /* Adams-Bashforth 2nd order step */
double *h,X[][smax],XD[][smax];
int *k;

{

int i;

for (i=1;i<=*k;i++)
  X[i][0]=fladd(X[i][1],flmult(*h/2.0,
fladd(flmult(3.,XD[i][1]),-XD[i][2])));

}

/****************************************************************************/

void GODSTEP(h,k,X,XDD) /* Godunov step */
double *h,X[][smax],XDD[][smax];
int *k;

{

int i;

for (i=1;i<=*k;i++)
X[i][0]=fladd(flmult(2.,X[i][1]),fladd(-X[i][2],
flmult(*h * *h,XDD[i][1])));

}

/****************************************************************************/

void EULSTEP(h,k,X,XD) /* Euler step */
double *h,X[][smax],XD[][smax];
int *k;

{

int i;

for (i=1;i<=*k;i++)
  X[i][0]=fladd(X[i][1],flmult(*h,XD[i][1]));

}

/****************************************************************************/

void LEAPSTEP(h,k,X,XD) /* Leapfrog step */
double *h,X[][smax],XD[][smax];
int *k;

{

int i;

for (i=1;i<=*k;i++)
  X[i][0]=fladd(X[i][2],flmult(2.0* *h,XD[i][1]));

}

/****************************************************************************/

void AB3STEP(h,k,X,XD) /* Adams-Bashforth 3rd order step */
```

Figure B.11 IVPSCHEMF.C

```
double *h,X[][smax],XD[][smax];
int *k;

{

int i;

for (i=1;i<=*k;i++)
  X[i][0]=fladd(X[i][1],flmult(*h/12.0,fladd(flmult(23.,XD[i][1]),
fladd(flmult(-16.,XD[i][2]),flmult(5.,XD[i][3])))));

}

/***************************************************************************/

void RK4STEP(f,h,k,X,XD,U) /* Runge-Kutta 4th order step */
void (*f)();
double *h,X[][smax],XD[][smax],U[];
int *k;

{

int i,j;
double XR[rmax][smax],XRD[rmax][smax];

(*f)(X,XD,U);
for (i=1;i<=*k;i++)
  {
  XRD[i][4]=XD[i][1];
  XR[i][3]=fladd(X[i][1],flmult(*h/2.0,XRD[i][4]));
  XR[i][1]=XR[i][3];
  }
(*f)(XR,XRD,U);
for (i=1;i<=*k;i++)
  {
  XRD[i][3]=XRD[i][1];
  XR[i][2]=fladd(X[i][1],flmult(*h/2.0,XRD[i][3]));
  XR[i][1]=XR[i][2];
  }
(*f)(XR,XRD,U);
for (i=1;i<=*k;i++)
  {
  XRD[i][2]=XRD[i][1];
  XR[i][1]=fladd(X[i][1],flmult(*h,XRD[i][2]));
  }
(*f)(X,XRD,U);
for (i=1;i<=*k;i++)
  X[i][0]=fladd(X[i][1],flmult(*h/6.0,
fladd(fladd(XRD[i][4],flmult(2.,XRD[i][3])),
fladd(flmult(2.0,XRD[i][2]),XRD[i][1]))));

}
```

Figure B.11 IVPSCHEMF.C

```
/* godanal.c */
/* to give analytic solutions for mechanical, electrical and hpde probs. */

/**********************************************************************/

void MECHAN(T,Y,YA,YEA,b)
double T[],Y[][cmax],YA[][cmax],YEA[][cmax],b;

{

double k1r[rmax],k1i[rmax],k2r[rmax],k2i[rmax],rr[rmax],ri[rmax],x1,x2;
int i=0;

k1r[1] =  -0.061671236895901;       k1i[1] =  -2.810700287201718E+07;
k1r[3] =  -0.014371386772571;       k1i[3] =   3.444521204947223E-17;
k1r[4] =   0.057729069111989;       k1i[4] =   0.190163686873597;
k1r[6] =  -0.050705313165149;       k1i[6] =  -0.004892921119899;
k1r[8] =  -7.205719968844154E-04;   k1i[8] =  -2.036547949608477E-20;
k1r[9] =   5.222234378871828E-04;   k1i[9] =  -8.667609909194935E-04;
k2r[1] =  -0.061671236895901;       k2i[1] =  -2.810700287201718E+07;
k2r[3] =  -0.144955398887593;       k2i[3] =   4.804292092426264E-17;
k2r[4] =   0.005315513533364;       k2i[4] =  -5.046440662086905E-04;
k2r[6] =   0.061359513411958;       k2i[6] =   0.018359057303193;
k2r[8] =   0.003085875899532;       k2i[8] =   1.148229183805980E-19;
k2r[9] =   0.004259734548708;       k2i[9] =   0.002040291509190;
rr[1] =   -6.379529120338866E-18;   ri[1] =    1.335487221920111E-09;
rr[3] =   -0.354279852653012;       ri[3] =    7.758076541149032E-26;
rr[4] =   -0.332576899489171;       ri[4] =    0.363802401052926;
rr[6] =   -0.902906986007217;       ri[6] =   -0.270023167886790;
rr[8] =   -3.094990471592307;       ri[8] =   -2.081668171172169E-17;
rr[9] =    0.0;                     ri[9] =    3.141592653589793;
while (T[i] < b)
  {
  x1 = k1r[3] * exp(rr[3] * T[i]) + k1r[8] * exp(rr[8] * T[i]) +
  2*exp(rr[1]*T[i])*(k1r[1]*cos(ri[1]*T[i])-k1i[1]*sin(ri[1]*T[i])) +
  2*exp(rr[4]*T[i])*(k1r[4]*cos(ri[4]*T[i])-k1i[4]*sin(ri[4]*T[i])) +
  2*exp(rr[6]*T[i])*(k1r[6]*cos(ri[6]*T[i])-k1i[6]*sin(ri[6]*T[i])) +
  2*exp(rr[9]*T[i])*(k1r[9]*cos(ri[9]*T[i])-k1i[9]*sin(ri[9]*T[i]));
  x2 = k2r[3] * exp(rr[3] * T[i]) + k2r[8] * exp(rr[8] * T[i]) +
  2*exp(rr[1]*T[i])*(k2r[1]*cos(ri[1]*T[i])-k2i[1]*sin(ri[1]*T[i])) +
  2*exp(rr[4]*T[i])*(k2r[4]*cos(ri[4]*T[i])-k2i[4]*sin(ri[4]*T[i])) +
  2*exp(rr[6]*T[i])*(k2r[6]*cos(ri[6]*T[i])-k2i[6]*sin(ri[6]*T[i])) +
  2*exp(rr[9]*T[i])*(k2r[9]*cos(ri[9]*T[i])-k2i[9]*sin(ri[9]*T[i]));
  YA[1][i] = x1 - x2;
  YEA[1][i] = fabs(YA[1][i] - Y[1][i]);
  i++;
  }

}

/**********************************************************************/

void ELECAN5(T,Y,YA,YEA,b) /* solution with r4=1/2 */
double T[],Y[][cmax],YA[][cmax],YEA[][cmax],b;

{

double kr[rmax],ki[rmax],rr[rmax],ri[rmax],x;
int i=0;

rr[1]=  -0.545536819194096;     ri[1]= 0.329561263141258;
rr[3]=  -0.332022027996957;     ri[3]= 0.923264468330181;
rr[5]=  -0.428516755789970;     ri[5]= -2.023367642157935;
rr[7]= -47.637848794037956;     ri[7]=  -5.551115123125783E-17;
```

Figure B.12 GODANAL.C

```
rr[8]= 0.0;                       ri[8]= 3.0;
kr[1]= 0.583447195445384;         ki[1]= 1.786846174581650;
kr[3]= -0.916092193431197;        ki[3]= -1.062135729773727;
kr[5]= 0.367460028418060;         ki[5]= -0.197269455199473;
kr[7]= -5.092417010178724E-09;  ki[7]= -2.892974149785409E-27;
kr[8]= -0.034815027886039;        ki[8]= -0.059661177389016;
while (T[i] < b)
  {
  x=kr[7]*exp(rr[7]*T[i])+2*exp(rr[1]*T[i])*(kr[1]*cos(ri[1]*T[i])-
       ki[1]*sin(ri[1]*T[i]))+2*exp(rr[3]*T[i])*(kr[3]*cos(ri[3]*T[i])-
       ki[3]*sin(ri[3]*T[i]))+2*exp(rr[5]*T[i])*(kr[5]*cos(ri[5]*T[i])-
       ki[5]*sin(ri[5]*T[i]))+2*exp(rr[8]*T[i])*(kr[8]*cos(ri[8]*T[i])-
       ki[8]*sin(ri[8]*T[i]));
  YA[1][i] = x;
  YEA[1][i] = fabs(YA[1][i] - Y[1][i]);
  i++;
  }

}

/*********************************************************************/

void ELECAN4(T,Y,YA,YEA,b) /* solution with r4=0 */
double T[],Y[][cmax],YA[][cmax],YEA[][cmax],b;

{

double kr[rmax],ki[rmax],rr[rmax],ri[rmax],x;
int i=0;

rr[1]= -0.416518822192015 ;      ri[1]= 0.286901247571467;
rr[3]=  -0.280959536111124;      ri[3]= -0.907053864275100;
rr[5]=  -0.104828707647075;      ri[5]= -2.163562679844431;
rr[7]= -47.645385868099570;      ri[7]=  0.000000000000000;
rr[8]= 0.0;                       ri[8]= 3.0;
kr[1]= -0.676542807200635;       ki[1]= -0.391754378830445;
kr[3]= 0.999958803345352;        ki[3]= 0.568965299607208;
kr[5]= -0.462256900894519;       ki[5]= -0.616478613594744;
kr[7]= .237204748228758e-6;      ki[7]= 0.000000000000000;
kr[8]= 0.138840786147429;        ki[8]= -0.218672044418416;
while (T[i] < b)
  {
  x=kr[7]*exp(rr[7]*T[i])+2*exp(rr[1]*T[i])*(kr[1]*cos(ri[1]*T[i])-
       ki[1]*sin(ri[1]*T[i]))+2*exp(rr[3]*T[i])*(kr[3]*cos(ri[3]*T[i])-
       ki[3]*sin(ri[3]*T[i]))+2*exp(rr[5]*T[i])*(kr[5]*cos(ri[5]*T[i])-
       ki[5]*sin(ri[5]*T[i]))+2*exp(rr[8]*T[i])*(kr[8]*cos(ri[8]*T[i])-
       ki[8]*sin(ri[8]*T[i]));
  YA[1][i] = x;
  YEA[1][i] = fabs(YA[1][i] - Y[1][i]);
  i++;
  }

}

/*********************************************************************/

void ELECANS(T,Y,YA,YEA,b) /* solution for small system */
double T[],Y[][cmax],YA[][cmax],YEA[][cmax],b;

{

double x;
int i=0;
```

Figure B.12 GODANAL.C

```
while (T[i] < b)
  {
  x=exp(-.5*T[i])*(3.287671*cos(.866*T[i])-2.37267234*sin(.866*T[i]))
    -3.287671*cos(3.0*T[i])+1.2329767*sin(3.0*T[i]);
  YA[1][i]=x;
  YEA[1][i]=fabs(YA[1][i]-Y[1][i]);
  i++;
  }

}

/**********************************************************************/

void EPDEAN(T,Y,YA,YEA,b)
double T[],Y[][cmax],YA[][cmax],YEA[][cmax],b;

{

/*double x[rmax][cmax];*/
int i=0,j;

while (T[i] < b)
  {
/*
  for (j=1;j<=10;j++)
    x[j][i]=sin(pi2*(double)j*dx)*cos(pi2*T[i]);
*/
  YA[1][i]=sin(pi2*1.)*cos(pi2*T[i]);
  YEA[1][i]=fabs(YA[1][i]-Y[1][i]);
  i++;
  }

}

/**********************************************************************/

void STABPEANAN(T,Y,YA,YEA,b)
double T[],Y[][cmax],YA[][cmax],YEA[][cmax],b;

{

int i=0;

while (T[i] <= b )
  {
  YA[1][i]=(exp(-sig*T[i])*sin(omeg*T[i]))/omeg;
  YEA[1][i]=fabs(YA[1][i]-Y[1][i]);
  i++;
  }

}

/**********************************************************************/

void SINGORAN(T,Y,YA,YEA,b)
double T[],Y[][cmax],YA[][cmax],YEA[][cmax],b;

{

int i=0,j;
double x;

while (T[i] <= b)
  {
```

Figure B.12 GODANAL.C

```
  for (j=1;j<=N-1;j++)
    {
    x=ab+dx*(double)j;
    YA[j][i]=4*atan(exp((x-.5*T[i])/pow(.75,.5)));
    YEA[j][i]=fabs(YA[j][i]-Y[j][i]);
    }
  i++;
  }

}

/**********************************************************************/

void ELECSMALLAN1(T,Y,YA,YEA,b)
double T[],Y[][cmax],YA[][cmax],YEA[][cmax],b;

{

int i=0;

while (T[i] <= b)
  {
  YA[1][i]=2.0*exp(-.5*T[i])*(0.006321108438466*cos(1.658312395177700*T[i])
  +0.329056688876620*sin(1.658312395177700*T[i]))+2.0*(-0.006321108438466*
  cos(9.424777960769380*T[i])-0.057562972190881*sin(9.424777960769380*T[i]));
  YEA[1][i]=fabs(YA[1][i]-Y[1][i]);
  i++;
  }

}

/**********************************************************************/

void ELECSMALLAN2(T,Y,YA,YEA,b)
double T[],Y[][cmax],YA[][cmax],YEA[][cmax],b;

{

int i=0;

while (T[i] <= b)
  {
  YA[1][i]=2.0*exp(-1.226698825758202*T[i])*(-0.415506500842584*
  cos(1.467711508710224*T[i])-0.172305600546402*
  sin(1.467711508710224*T[i]))+2.0*(0.011381952218377*
  cos(9.424777960769380*T[i])-0.003810275057188*sin(9.424777960769380*T[i]))
  +0.808249097248413*exp(-0.546602348483596*T[i]);
  YEA[1][i]=fabs(YA[1][i]-Y[1][i]);
  i++;
  }

}

/**********************************************************************/

void ELECSMALLAN3(T,Y,YA,YEA,b)
double T[],Y[][cmax],YA[][cmax],YEA[][cmax],b;

{

int i=0;

while (T[i] <= b)
  {
```

Figure B.12 GODANAL.C

```
    YA[1][i]=2.0*exp(-0.5*T[i])*(0.530448482176386*
    cos(0.866025403784439*T[i])-0.313228660033139*
    sin(0.866025403784439*T[i]))+2.0*(-0.530448482176386*
    cos(9.424777960769380*T[i])+0.056923167856408*sin(9.424777960769380*T[i]));
    YEA[1][i]=fabs(YA[1][i]-Y[1][i]);
    i++;
    }

}

/***************************************************************************/

void ELECSMALLAN4(T,Y,YA,YEA,b)
double T[],Y[][cmax],YA[][cmax],YEA[][cmax],b;

{

int i=0;

while (T[i] <= b)
    {
    YA[1][i]=2.0*exp(-0.75*T[i])*(0.004516224340959*
    cos(0.968245836551854*T[i])+0.275044848789804*
    sin(0.968245836551854*T[i]))+2.0*(-0.004516224340959*
    cos(9.424777960769380*T[i])-0.027897088137722*sin(9.424777960769380*T[i]));
    YEA[1][i]=fabs(YA[1][i]-Y[1][i]);
    i++;
    }

}
```

Figure B.12 GODANAL.C

APPENDIX C - ANALYTIC SOLUTION DERIVATIONS

1) The Wave Equation ($\frac{\partial^2 U}{\partial t^2} = \frac{\partial^2 U}{\partial x^2}$).

The general form of the solution to the wave equation is $U(x,t) = f(x-t) + g(x+t)$. The functions f and g can be found by using the initial conditions and the boundary conditions of the problem. Initial conditions of

$$U(x,0) = sin\frac{\pi}{2}x \quad ,$$
$$\frac{\partial U(x,0)}{\partial t} = 0 \quad ,$$

and boundary conditions of

$$U(0,t) = 0 \quad ,$$
$$\frac{\partial U(1,t)}{\partial x} = 0 \quad ,$$

were used for this example. f and g are found by substituting the general solution into the initial conditions and boundary conditions:

$$U(x,0) = f(x) + g(x) = sin\frac{\pi}{2}x \quad \text{so} \quad f(x) = sin\frac{\pi}{2}x - g(x) \qquad (C.1)$$
$$\frac{\partial U(x,0)}{\partial t} = -f'(x) + g'(x) = 0 \quad \text{so} \quad f'(x) = g'(x) \qquad (C.2)$$
$$U(0,t) = f(-t) + g(t) = 0 \quad \text{so} \quad f(-t) = -g(t) \qquad (C.3)$$
$$\frac{\partial U(1,t)}{\partial x} = f'(1-t) + g'(1+t) = 0 \quad . \qquad (C.4)$$

Differentiating $(C.1)$ and then substituting into $(C.2)$:

$$f'(x) = \frac{\pi}{2}cos\frac{\pi}{2}x - g'(x) = g'(x)$$
$$\text{or} \quad \frac{\pi}{4}cos\frac{\pi}{2}x = g'(x)$$
$$\text{so that} \quad g(x) = \frac{1}{2}sin\frac{\pi}{2}x + c \quad . \qquad (C.5)$$

Substituting $(C.5)$ back into $(C.1)$ gives:

$$f(x) = sin\frac{\pi}{2}x - \frac{1}{2}sin\frac{\pi}{2}x - c = \frac{1}{2}sin\frac{\pi}{2}x - c \quad .$$

Since f and g have been found using only the initial conditions, we need to check that they satisfy the boundary conditions and the differential equation. We can check the $x = 0$ boundary condition by substituting f and g into $(C.3)$,

$$f(-t) = \frac{1}{2}sin(-\frac{\pi}{2}t) - c = -\frac{1}{2}sin\frac{\pi}{2}t - c = -g(t) \quad ,$$

and we can check the other boundary condition by substituting f and g into $(C.4)$ so that

$$\begin{aligned}
&f'(1-t) + g'(1+t) \\
&= \frac{\pi}{4}cos\frac{\pi}{2}(1-t) + \frac{\pi}{4}cos\frac{\pi}{2}(1+t) \\
&= \frac{\pi}{4}[cos\frac{\pi}{2}cos\frac{pi}{2}t + sin\frac{\pi}{2}sin\frac{\pi}{2}t] + \frac{\pi}{4}[cos\frac{\pi}{2}cos\frac{\pi}{2}t - sin\frac{\pi}{2}sin\frac{\pi}{2}t] \\
&= 0 \quad .
\end{aligned}$$

We check to see that f and g satisfy the differential equation by substituting into the general solution and differentiating: *

$$\begin{aligned}
U(x,t) &= \frac{1}{2}[sin\frac{\pi}{2}(x-t) + sin\frac{\pi}{2}(x+t)] \\
\frac{\partial U(x,t)}{\partial t} &= \frac{\pi}{4}[-cos\frac{\pi}{2}(x-t) + cos\frac{\pi}{2}(x+t)] \\
\frac{\partial^2 U(x,t)}{\partial t^2} &= -\frac{\pi^2}{8}[sin\frac{\pi}{2}(x-t) + sin\frac{\pi}{2}(x+t)] \\
\frac{\partial U(x,t)}{\partial x} &= \frac{\pi}{4}[cos\frac{\pi}{2}(x-t) + cos\frac{\pi}{2}(x+t)] \\
\frac{\partial^2 U(x,t)}{\partial t^2} &= -\frac{\pi^2}{8}[sin\frac{\pi}{2}(x-t) + sin\frac{\pi}{2}(x+t)] = \frac{\partial^2 U(x,t)}{\partial t^2} \quad .
\end{aligned}$$

* We can let c be equal to zero since it does not affect either the initial conditions, boundary conditions or the differential equation.

2) The Mechanical Model of the Human Body.

Applying Newton's Law to the system of Figure (2.2) results in four $2nd$ order differential equations:

$$\begin{aligned}
&\ddot{x}_1 + \frac{b_1}{m_1}\dot{x}_1 + \frac{k_1}{m_1}x_1 - \frac{b_1}{m_1}\dot{x}_2 - \frac{k_1}{m_1}x_2 = 0\\
&-\frac{b_1}{m_2}\dot{x}_1 - \frac{k_1}{m_2}x_1 + \ddot{x}_2 + \frac{b_1+b_2+b_3}{m_2}\dot{x}_2\\
&+\frac{k_1+k_2+k_3}{m_2}x_2 - \frac{b_2}{m_2}\dot{x}_3 - \frac{b_3}{m_2}\dot{x}_4 - \frac{k_3}{m_2}x_4 = 0\\
&-\frac{b_2}{m_3}\dot{x}_2 - \frac{k_2}{m_3}x_2 + \ddot{x}_3 + \frac{b_2}{m_3}\dot{x}_3 + \frac{k_2}{m_3}x_3 = 0\\
&-\frac{b_3}{m_4}\dot{x}_2 - \frac{k_3}{m_4}x_2 + \ddot{x}_4 + \frac{b_3}{m_4}\dot{x}_4 + \frac{k_3}{m_4}x_4 = \frac{1}{m_4}u
\end{aligned}$$

where $x_1 = d_1$, $x_2 = d_2$, $x_3 = d_3$, $x_4 = d_4$ and $u = f = 10sin\pi t$. Since initial conditions are chosen to be zero, the use of Laplace Transforms on the above differential equations will result in the equation:

$$\begin{pmatrix}
s^2 + \frac{b_1}{m_1}s + \frac{k_1}{m_1} & -\frac{b_1}{m_1}s - \frac{k_1}{m_1} & 0 & 0\\
-\frac{b_1}{m_2}s - \frac{k_1}{m_2} & s^2 + \frac{b_1+b_2+b_3}{m_2}s + \frac{k_1+k_2+k_3}{m_2} & -\frac{b_2}{m_2}s - \frac{k_2}{m_2} & -\frac{b_3}{m_2}s - \frac{k_3}{m_2}\\
0 & -\frac{b_2}{m_3}s - \frac{k_2}{m_3} & s^2 + \frac{b_2}{m_3}s + \frac{k_2}{m_3} & 0\\
0 & -\frac{b_3}{m_4}s - \frac{k_3}{m_4} & 0 & s^2 + \frac{b_3}{m_4}s + \frac{k_3}{m_4}
\end{pmatrix}$$

$$\begin{pmatrix} X_1(s)\\ X_2(s)\\ X_3(s)\\ X_4(s)\end{pmatrix} = \begin{pmatrix} 0\\ 0\\ 0\\ \frac{U(s)}{m_4}\end{pmatrix} .$$

$X_1(s)$ and $X_2(s)$ can be solved for by the use of determinants and then expressions for $x_1(t)$ and $x_2(t)$ can be found immediately by the use of partial fraction expansions and then inverse Laplace Transforms on $X_1(s)$ and $X_2(s)$. This leads to an expression for the output,

$$\begin{aligned}
y(t) = x_1(t) - x_2(t) = {}& .1306e^{-.35431t} - .0038e^{-3.095t}\\
&+ 2e^{-.3326t}(.0524cos(.3638t) - .19067sin(.3638t))\\
&- 2e^{-.903t}(.112cos(.27t) + .02325sin(.27t))\\
&+ 2(-.00374cos(\pi t) + .0029sin(\pi t))
\end{aligned}$$

3) The $7th$ Order Passive Electrical System.

Application of Kirchoff's Current Law to each of the five nodes of the circuit of Figure (2.3) results in five integro-differential equations:

$$\begin{aligned}
&C_1\frac{dv_1(t)}{dt}+\frac{v_1(t)}{R_1}+\frac{1}{L_1}\int(v_1(t)-v_2(t))\,dt=i_s(t)\\
&C_2\frac{dv_2(t)}{dt}+\frac{v_2(t)}{R_2}+\frac{1}{L_2}\int(v_2(t)-v_3(t))\,dt+\frac{1}{L_1}\int(v_2(t)-v_1(t))\,dt=0\\
&C_3\frac{dv_3(t)}{dt}+\frac{v_3(t)-v_4(t)}{R_3}+\frac{1}{L_2}\int(v_3(t)-v_2(t))\,dt=0\\
&C_4\frac{dv_4(t)}{dt}+\frac{v_4(t)-v_3(t)}{R_3}+\frac{1}{L_3}\int(v_4(t)-v_5(t))\,dt=0\\
&\frac{v_5(t)}{R_4}+\frac{1}{L_3}\int(v_5(t)-v_4(t))\,dt=0\quad .
\end{aligned}$$

Using initial conditions of zero and Laplace Transforms on the above equations will result in an equation in the s-domain similar to that found for the mechanical system but omitted here due to lack of space. Using the same technique as was used for the mechanical system, an expression for $v_5(t)$ is then easily found by using the method of determinants, then partial fraction expansions and then by the use of inverse Laplace Transforms. An expression for the output, $v_5(t)$, is then given by

$$\begin{aligned}
v_5(t)=-5.0910^{-9}e^{-47.64t}&+2e^{-.4285t}(.3675cos2.0234t-.197sin2.0234t)\\
&+2e^{-.332t}(-.9161cos.9233t+1.0621sin.9233t)\\
&+2e^{-.5455t}(.5834cos.3296t-1.7868sin.3296t)\\
&+(-.0348cos3t+.0597sin3t)\quad .
\end{aligned}$$

REFERENCES

[1] Cooke, C.H., "On Shock Capturing by First and Second Order Accurate Godunov Methods", Mathematics and Computers in Simulation, Vol.26, pp.485-496.

[2] Eidelman, S. et al. (1984), "Application of the Godunov Method and Its Second-Order Extension to Cascade Flow Modeling". AIAA Journal, Vol.22, No.11, pp.1609-1615.

[3] Godunov, S.K. et al., "Finite Difference Methods for Numerical Computations of discontinuous solutions of equations of fluid dynamics", Mat. Sb. 47, 1959, pp.271-295. (In Russian), Cornell Aeronautical Lab. Trans.

[4] Godunov, S.K. et al., "Numerical Solution of Multidimensional Problems in Gas Dynamics", Nauka, Moscow, 1976 (in Russian).

[5] Harten, A. et al (1983), "On Upstream Differencing and Godunov-Type Schemes for Hyperbolic Conservation Laws", SIAM Review, Vol.25, No.1, pp. 35-61.

[6] Hostetter, Savant Jr., Stafini, "Design of Feedback Control Systems", (1989), 2nd Edition Saunders College Publishing

[7] Lambert, J.D., "Computational Methods in Ordinary Differential Equations", (1973), John Wiley

[8] Leveque, R.J. (1984), "Convergence on a Large Time Step Generalization of Godunov's Method for Conservation Laws", Comm. on Pure and Applied Mathematics, Vol. 37, pp.463-477.

[9] Leveque, R.J. (1985), "A Large Time Step Generalization of Godunov's Method for Systems of Conservation Laws", SIAM Journal of Numerical Analysis, Vol. 22, No. 6, pp.1051-1075.

[10] Lucier, B.J. (1985), "Error Bounds for the Methods of Glimm, Godunov and LeVeque", SIAM Journal of Numerical Analysis, Vol. 22, No. 6, pp.1074-1081.

[11] Mathworks, Inc. (1987), MATLAB with System Identification Toolbox and Control System Toolbox - User Manual,
21 Eliot St., South Natick, MA 01760.

[12] Scott, A.C. et al, "The Soliton: A New Concept in Applied Science", Proceedings of the IEEE, Oct. 1973

[13] Systems Control Technology, Inc. (1985), CTRL-C, A Language for the Computer-Aided Design of Multivariable Control Systems,
User's Guide, 2300 Geng Rd., P.O. Box 10180, Palo Alto, CA 94303.