SIMULATION OF PARTITIONED SYSTEMS USING AVERAGING TECHNIQUES FOR COUPLING VARIABLES

by

Lajoo Narain Motwani

A Thesis Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

MASTER OF SCIENCE

In the Graduate College

THE UNIVERSITY OF ARIZONA

1 9 7 9

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 judgment 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: Lajoo Motwani

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

Olgierd A. Palusinski

O. A. PALUSINSKI
Visiting Professor of
Electrical Engineering

May 31, 1979
Date

ACKNOWLEDGEMENTS

This research was supported by National Science Foundation's Grant ENGR77-01431. Thanks are given to Dr. O. A. Palusinski and Dr. J. V. Wait for their supervision and to Dr. R. H. Mattson for providing the facilities to carry out the research.

TABLE OF CONTENTS

TABLE OF CONTENTS--Continued

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS--Continued

ABSTRACT

This thesis describes a study of partitioned system integration algorithms which use averaging of variables at the interface between a fast subsystem and a slower subsystem. The algorithms were coded for the DAREP continuous system simulation language. Two so-called combined algorithms are useful when the fast subsystem is linear (the slow subsystem may be nonlinear). Other algorithms studied are valid in the case when both subsystems are nonlinear.

The algorithms were tested by simulating several partitioned systems and the results were compared to simulations done with conventional partitioned algorithms not employing averaging. It was found that averaging improved worst-case peak fractional errors for larger step sizes for the experiments, but as expected, the mean peak error was found to be problem dependent. In addition, execution times, and thus, costs were improved when using the combined algorithms, but both nonlinear algorithms required longer execution times, and therefore, higher costs.

CHAPTER 1

INTRODUCTION

This thesis reports on the coding and testing of four integration algorithms employing averaging techniques to be used in the DAREP (Lucas and Wait, 1975) package for the simulation of partitioned systems (Palusinski and Wait, 1978). The general class of partitioned systems referred to here consist of those which may be divided into one fast and one slow continuous-time subsystem. The four algorithms take advantage of this property by using a large step size and an average of the appropriate fast subsystem variables over this large step size in the integration of the slow subsystem.

The first two alrogithms considered were taken from thoeritical work originally presented in Palusinski (1977a) and were intended for use with systems having a slow subsystem which is nonlinear and a fast subsystem which is linear. These two algorithms will be referred to as the combined methods. The two remaining algorithms, labelled the partitioned methods, were derived from Palusinski (1978) to be used with a fast and a slow nonlinear subsystem.

The strategy utilized in the coding of these simulation methods consisted of writing FORTRAN subroutines which would be compatible with the DAREP simulation language. This resulted in each algorithm being expressed as an integration subroutine, named INTRGX, containing expressions for the computation of the next states of both subpartitions, together with several accompanying subprograms performing initialization, etc. The combined methods required extensive matrix manipulations due to the description of the linear subsystem in matrix form. These manipulations were performed with the aid of a library of matrix subroutines (Ferguson, 1972).

After programming, the first combined method was tested on three problems: a sine loop (harmonic oscillator) problem, the simulation of a servo-controlled pendulum, and the simulation of a mine-shaft elevator. A second combined method was tested on the servo-controlled pendulum. The partitioned methods were tested on the models of a nonlinear electronic oscillator (two different partitions) and an autopilot hydraulic servo-system. In all cases, the results were compared to previous partitioned system algorithms that do not use averaging.

The organization of this thesis is as follows: Chapter 2 examines the notation and conventions used and describes standard variable names and assumptions. Chapter 3 covers the combined methods including a general description of each algorithm together with the special considerations of coding and use. Chapter 4 presents the partitioned methods. Chapter 5 is devoted to the description of the test problems, and overall results and conclusions are discussed in Chapter 6. The program listings and detailed results have been placed in the appendices.

CHAPTER 2

SYSTEM DESCRIPTION: NOTATION

The description of the integration rules under consideration in the form of next-state equations requires a discussion of the notation. This notation arises from modeling the general partitioned systems.

The general class of partitionable systems of interest here are assumed to be composed of a fast and a slow subsystem as shown in Figure 1.

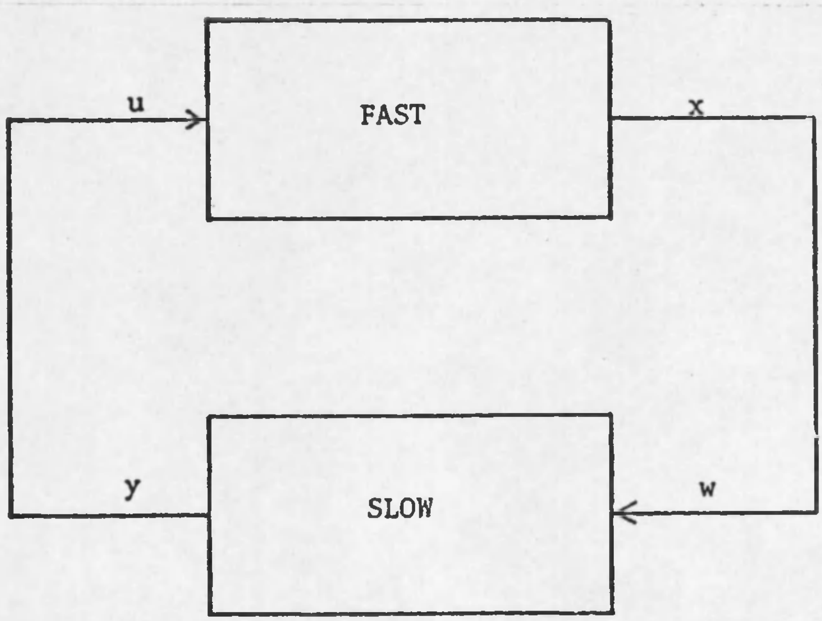

Figure 1. Block diagram showing partitioned system

The slow nonlinear subsystem may be represented by state differential equations (Palusinski, 1977a, p. 1)

$$\dot{y} = f\ (y,\ t,\ w) \tag{2.1}$$

with initial state y (0) and output coupling equation

$$u = g\ (y,\ t) \tag{2.2}$$

where: y - state vector of dimension k_1

u - output vector of dimension k_2

w - input vector of dimension k_3

t - independent variable

Of course, additional output variables may be obtained from each subsystem, but only those which interconnect the two regions are of interest here.

Combined Linear-Nonlinear Case

Here the fast subsystem is characterized by linear state differential equations (Palusinski, 1977a, p. 2)

$$\dot{x} = Ax + Bu \tag{2.3}$$

and output coupling equation

$$w = Cx + Du \tag{2.4}$$

where: x - state vector of dimension k_4

u, w - input and output vectors as before

A - constant state matrix

B - constant input matrix

C, D - constant output matrix

Thus the linear system is denoted by the A, B, C, and D matrices, the initial conditions $x(0)$, and the input u.

Partitioned Nonlinear Case

In this case it is assumed that both the fast and slow subsystems are nonlinear. The slow nonlinear subsystem is described again by the state differential equation (2.1) with w = x and (2.2) and the fast nonlinear subsystem by another set of differential equations (Palusinski, 1978, p. 1)

$$\dot{x} = f_1 (x, t, y) \tag{2.5}$$

with initial state $x(0)$.

where: x - state vector of dimension l_1

y - input coupling vector of dimension l_2

t - independent variable

Time Discretization

A simulation is assumed to begin at $t = 0.0$. Values of the system variables are then calculated at equally-spaced time intervals, $t_n = nh$, $n = 0, 1, 2, \ldots$ In some cases, values of the system variables are calculated at intermediate times and may also be output. The value of a variable x at $t = t_n$ is denoted by x_n, and x at $t = t_n + qh$ by x_{n+q} where q has a value between zero and one.

Slow Subsystem Discretization

The slow subsystem discretization technique (Palusinski, 1977a, pp. 3-6) used by all the algorithms is derived from equation (2.1) written in the form

$$y_{n+1} - y_n = \int_{t_n}^{t_{n+1}} f\ (y,\ t,\ w)\ dt \qquad (2.6)$$

The variable w may be represented in the interval t_n, t_{n+1} by an average value plus a variation:

$$w = \bar{w}_n + w \qquad (2.7)$$

where $\bar{w}_n = \frac{1}{h} \int_{t_{n+1}}^{t_n}$ wdt and w is the variation in w. Taking into account equation (2.7), it is possible to develop a Taylor series around $\bar{w}_n$. This tranforms equation (2.6) into

$$y_{n+1} - y_n = {}_{t_n}^{t_{n+1}}\ f\ (y,\ t,\ \bar{w}_n)\ dt + \int_{t_n}^{t_{n+1}} F_n.\ \delta w + 0\ (11\delta w11^2)\ dt \qquad (2.8)$$

wjere the matrix F_n is given by

$$F_n = \frac{\partial f\ (y,\ t,\ w)}{\partial w} \Bigg|_{\ w\, =\, \bar{w}_n} \qquad (2.9)$$

Averaging the Fast Linear Subsystem

The technique (Palusinski, 1977a, pp. 12-17) used by the combined linear-nonlinear algorithms in the averaging of the linear system is based on an equation (2.3) which is used to derive

$$x_{n+} = e^{A\delta} x_n + \int_0^{\delta} e^{A(\delta\ -\ \sigma\ )}\ Bu_{n+\delta}\ d\sigma \qquad (2.10)$$

where $0 \leq \delta \leq$ h. From this, an average value of x over the interval $\left[t_n,\ t_{n+1}\right]$ given by

$$\bar{x}_{n+1} = \frac{1}{h} \int_0^h x_{n+\sigma} \, d\sigma \tag{2.11}$$

may be computed by integrating both sides of equation (2.10) and replacing

$$u_{n+s} = 0 = 1 \quad + \; 2\,(\delta/h)^2 \; + \ldots + \; (\delta/h)^1 \tag{2.12}$$

The vectors are the linear combinations of the given values u_{n+i}. Following some manipulation, $\bar{x}_{n+1}$ may be defined by

$$\bar{x}_{n+1} = (AV_0 + I) \; x_n + \sum_{i=0}^{1} V_i \, B\lambda_i \tag{2.13}$$

where the matrices V_i are computed as follows

$$V_j = j! \sum_{k=j+1}^{\infty} \frac{A^{k-j-1}}{(k+1)!} \qquad h^{k-j} \tag{2.14}$$

$$V_{i-1} = \frac{h}{i} \; (AV_i + \frac{1}{i+1} \qquad I) \tag{2.15}$$

and $i = j, j-1, j-2, \ldots, 1$. The matrices, M_1, used in the combined linear nonlinear algorithms are related to V_i by

$$M_i = i \; V_{i-1} \tag{2.16}$$

Averaging of the Fast Nonlinear Subsystem

The partitioned nonlinear algorithm averages the fast subsystem variables as follows (Palusinski, 1978, p. 2).

$$\bar{x}_n = \sum_{i=1} A_i \; x_{n+a_i} \tag{2.17}$$

where a_i $(i = 1, 2, \ldots, n)$ are the variable step sizes used in the integration of the fast subsystem. The fractions a_i have to satisfy the relation

$$\sum_{i=1}^{n} a_i = 1 \qquad (2.18)$$

The preceding discussion has described the background information and underlying assumptions essential to the understanding of the mathematical representation of the simulation methods in subsequent chapters.

CHAPTER 3

DESCRIPTION OF COMBINED ALGORITHMS

The description of the two combined algorithms is divided into three parts: presentation of the algorithms, programming conventions, and the use of the programs.

Mathematical Representation of Algorithms

The first combined linear-nonlinear algorithms investigated is based on the Improved Euler equation (Palusinski, 1977a, pp. 7-8). The nonlinear system integration is performed by first computing derivatives k_1 and k_2 at t_n and t_{n+1} as shown in the following equations.

$$k_1 = f\ (y_n,\ t_n,\ \bar{w}_n) \tag{3.1}$$

$$k_2 = f\ (y_n +\ \ k_1,\ t_{n+1},\ \bar{w}_n) \tag{3.2}$$

Next, the value u_{n+1} is computed using k_1 and k_2 as follows

$$y_{n+1} = y_n + (h\ /2)\ (k_1 + k_2) \tag{3.3}$$

This results in a nonlinear output given by

$$u_{n+1} = g\ (y_{n+1},\ t_{n+1}) \tag{3.4}$$

The linear system is based upon the following equations (Palusinski, 1977a, pp. 24-25).

$$\bar{x}_{n+1} = (AV_0 + I)\ x_n + (V_0 - V_1)\ Bu_n + V_1Bu_{n+1} \tag{3.5}$$

$$\bar{w}_{n+1} = C\bar{x}_{n+1} + 1/2D\ (u_n + u_{n+1}) \tag{3.6}$$

The discrete value, x_{n+1}, of the linear system and the output w_{n+1} are computed as follows

$$x_{n+1} = (AM_0 + I)\ x_n + (M_0 - M_1)\ Bu_n + M_1Bu_{n+1}$$

$$w_{n+1} = Cx_{n+1}\ Du_{n+1} \tag{3.7}$$

The algorithm is completed with the nonlinear system correction

$$y^c{}_{n+1} = y_{n+1} + hF_n\ (\bar{w}_{n+1} - \bar{w}_n) \tag{3.8}$$

Modified Euler Algorithm

The second combined algorithm is derived from the Modified Euler Equation (Palusinski, 1977a, pp. 7-8). Here, solutions to the nonlinear system are first computed at the half step interval (i.e., $t_{n+1/2}$) as shown below (Palusinski, 1977a, pp. 21-22).

$$k_1 = f\ (y_n,\ t_n,\ \bar{w}_n) \tag{3.9}$$

$$y_{n+1/2} = y_n + (h/2)k_1 \tag{3.10}$$

$$u_{n+1/2} = g\ (y_{n+1/2},\ t_{n+1/2}) \tag{3.11}$$

This leads to a full step computation achieved by

$$k_2 = f\ (y_{n+1/2},\ t_{n+1/2},\ \bar{w}_n) \tag{3.12}$$

$$y_{n+1} = y_{n+1/2} + (h/2)\ k_2 \tag{3.13}$$

$$y_{n+1} = g\ (y_{n+1},\ y_{n+1}) \tag{3.14}$$

The linear system averaging is computed as follows (Palusinski, 1977a, p. 23).

$$\bar{x}_{n+1} = (AV_0 + I)\ x_n + (V_0 - 3V_1 + 2V_2)\ Bu_n + 4(V_1 - V_2)\ Bu_{n+1/2} + (2V_q - V_1)\ Bu_{n+1} \tag{3.15}$$

$$\bar{w}_{n+1} = \bar{x}_{n+1} + (D/6)\ (u_n + 4u_{n+1/2} + u_{n+1}) \tag{3.16}$$

The linear full step solution is given by

$$x_{n+1} = (AM_0 + I)\ x_n + (M_0 - 3M_1 + 2M_2)Bu_n + 4(M_1 - M_2)Bu_{n+1/2} + (2M_2 - M_1)Bu_{n+1} \tag{3.17}$$

Finally, the nonlinear system correction has the form

$$y^c_{n+1} = y_{n+1} + hF_n\ (\bar{w}_{n+1} - \bar{w}_n) \tag{3.18}$$

Programming Conventions

The coding of the above simulation methods was governed by the requirement that both programmed algorithms had to be compatible with DAREP. This resulted in the implementation of each integration rule as a FORTRAN subroutine named INTRGX. It was found necessary to include in each program two more subroutines named INICON and LINKW. In the case of the Modified Euler method, the matrix library subroutine MXCAL (Ferguson, 1972) was modified to suit the algorithm.

These are two major functions performed by the subroutine INTRGX. The first is the computation of the state-spaced and other matrices needed in the averaging and solution of the linear system.

These include M_1, V_1 e^{At}, etc. The execution of this portion of INTRGX is performed only once during each run at t = 0.0 and constitutes the precomputation or initialization section of the subroutine. The main part of the subroutine is associated with actual solution computation. Each time INTRGX is called, all state variables, defined variables, and average values are updated from t_n to t_{n+1}. Due to the matrix representation of the linear subsystem, extensive matrix manipulations are involved. The matrix operations are implemented using a library of subroutines available on permanent file (Ferguson, 1972).

The main purpose of INICON, is to compute initial conditions. This is done by first zeroing all matrices and vectors--an especially useful feature in multiple run simulations. Next, user defined subroutines are called to initialize the A, B, C, D, and F matrices. Finally, the initial values of the u, w, and $\bar{w}$ vectors are calculated.

The subroutine LINKW links the linear subsystem variables and averages needed in the derivative block to the values computed in INTRGX. This type of arrangement is necessary to avoid searching through the undefined parameter array created by DAREP to determine which elements of the array correspond to the proper linear variables.

In addition to the coding of these three subroutines for each combined linear nonlinear method, the algorithm based on the Modified Euler equation required the alteration of subroutine MXCAL from the subroutine library. The alteration was needed for the computation of matrices M_3 and V_2 which are used in that algorithm. A short description of the changes made to MXCAL appears in Appendix B and a listing of the new routine, MXCAL3, may be seen as part of the listing for the Modified Euler Method (Appendix A).

In all subroutines coded, care was taken to assign variable names according to those used in the next-state equations. This feature is seen in the program listings of the algorithms in Appendix A. As seen from these listings, subroutines INICON and LINKW are very straightforward and therefore no flow charts for these appear. Flow charts for the INTRGX subroutines for the two combined algorithms are found in Figures 2 and 3.

Use of Combined Linear-Nonlinear Algorithms: An Example

The simulation of a servo-controlled pendulum (Palusinski and Wait, 1978, pp. 14-16) by means of the Improved Euler based method was performed. The description is shown in Appendix A. In this example, it is found that apart from the three subroutines described previously, all the code shown is a user supplied description in accordance with Chapter 2.

The \$D1 block contains the differential equations corresponding to the slow nonlinear portion of the pendulum and a procedure section which calls LINKW (the \$D1 characteristics, as those of other DAREP blocks, are described in Korn and Wait, 1977). This call to LINKW updates averages of the linear variables needed by the nonlinear subsystem and values of w needed for output. The specifications of the nonlinear system is completed by the subroutine GFUNC which computes the output function u according to equation (2.2).

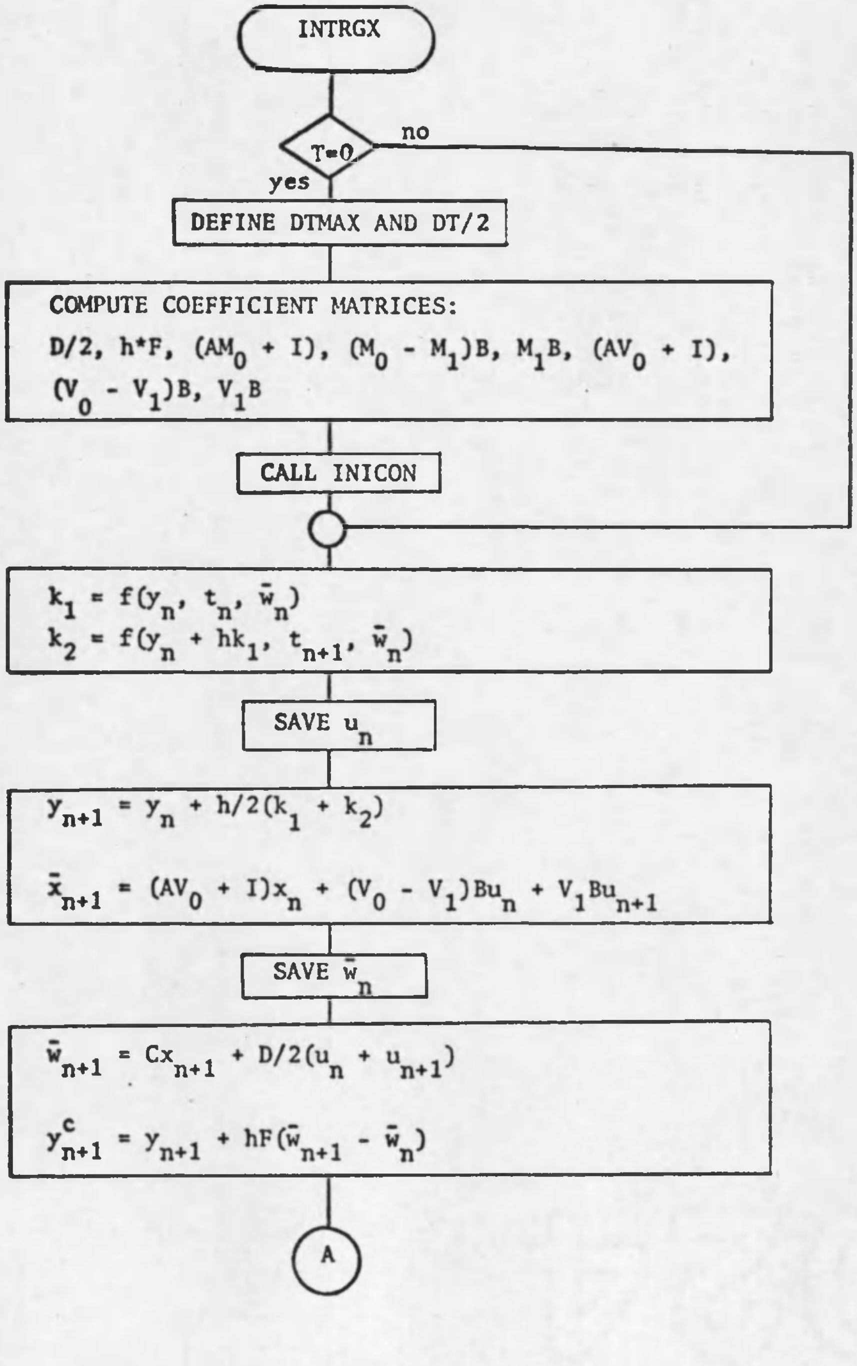

Figure 2. Flow chart of subroutine INTRGX
for Improved Euler method

A

$$x_{n+1} = (AM_0 + I)x_n + (M_0 - M_1)Bu_n + M_1Bu_{n+1}$$

$$w_{n+1} = Cx_{n+1} + Du_{n+1}$$

NEXT STEP PREPARATION

RETURN

Figure 2--Continued

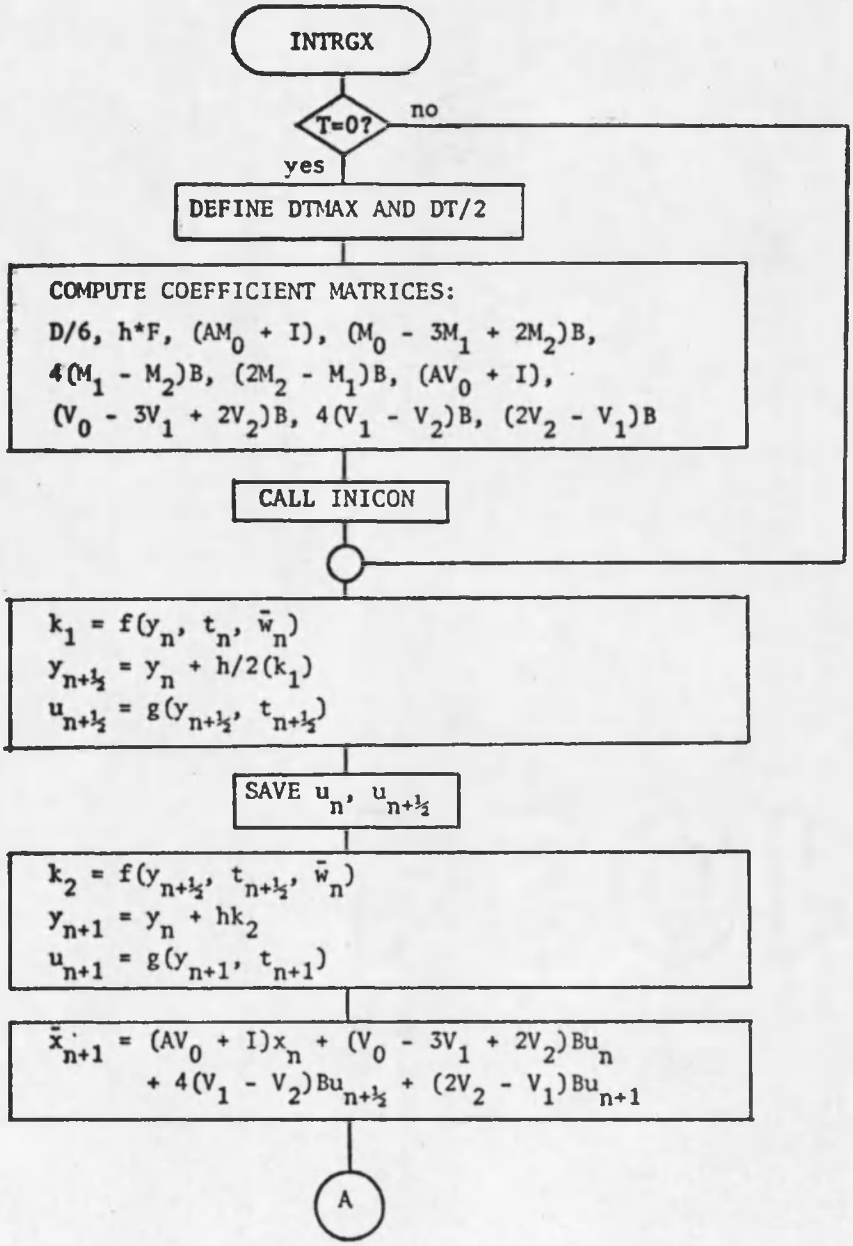

Figure 3. Flow chart of INTRGX for Modified Euler method

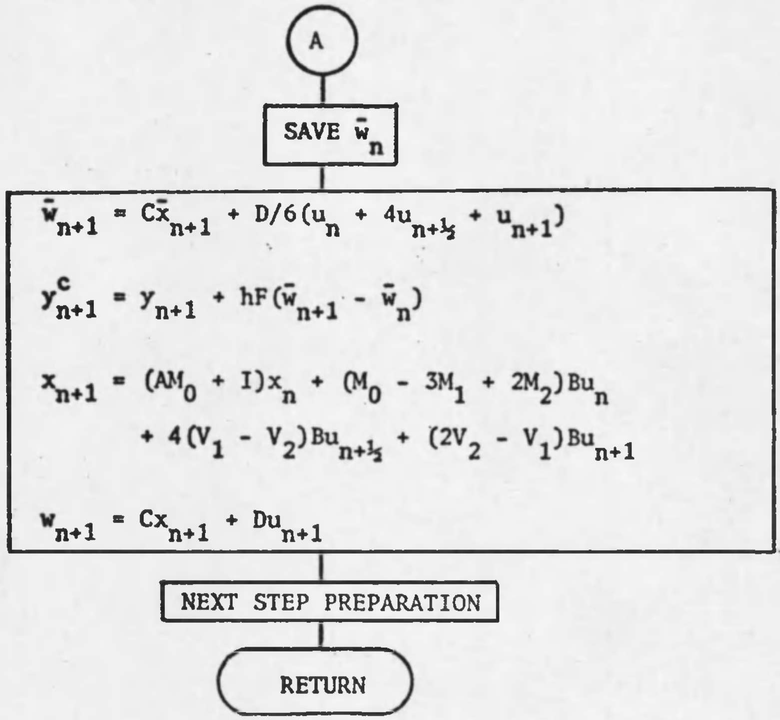

Figure 3--<u>Continued</u>

The linear subsystem description is provided in DEFLIN which defines the A, B, C, D, and F matrices, the initial conditions on x and the number of variables that are to be linked.

The two subroutines GFUNC and DEFLIN, are placed in the $F block followed by the algorithm subroutines INTRGX, INICON, and LINKW in the $0 block. Finally, initial conditions on the nonlinear system and output requests are entered.

By following the procedure outlined below, the user may simulate any linear-nonlinear partitioned system with the two combined algorithms. It is noted here that the Modified Euler based algorithm requires the inclusion of MXCAL3 in the $0 block.

CHAPTER 4

DESCRIPTION OF PARTITIONED ALGORITHMS

The description of the general nonlinear algorithms follows the format used in the previous chapter, viz., the presentation of the algorithms followed by programming conventions and an example of program use.

Mathematical Representation of the Nonlinear Algorithms

General Nonlinear Algorithm with Averaging

The first general nonlinear algorithm studied (Palusinski, 1978) employs a fixed step four point Runge-Kutta integration rule to compute solutions to the slow subsystem and a variable step Runge-Kutta Merson method to integrate the fast system (Korn and Wait, 1977, Appendix A). Averages of the fast subsystem variables are computed over the half step intervals t_n, $t_{n+1/2}$ and $t_{n+1/2}$, t_{n+1} as follows

$$\bar{x}_{n+1/2} = \sum_{i=i}^{m} a_i; \quad x_{n+\bar{a}_i} \tag{4.1}$$

$$\bar{x}_{n+1} = \sum_{i=i}^{m} b_1 . \quad x_{n+1/2+\bar{b}_1} \tag{4.2}$$

where the fractions a_i and b_i are constrained by

$$\sum_{i=i}^{m} a_i = 1/2 \tag{4.3}$$

$$\sum_{i=1}^{m} b_i = 1 \tag{4.4}$$

and $(h/2)a_i$ and $(h/2)b_i$ are the variable step sizes used in the first and second half steps. These averages are then used in the solution of the slow subsystem variables as seen here.

$$k_1 = f\,(y_n,\ t_n,\ \bar{x}_n) \tag{4.5}$$

$$k_2 = f\,(y_n + h/2k_1,\ t_{n+1/2},\ \bar{x}_{n+1/2}) \tag{4.6}$$

$$k_3 = f\,(y_n + (h/2)k_2,\ t_{n+1/2},\ \bar{x}_{n+1/2}) \tag{4.7}$$

$$y_{n+1/2} = y_n + (h/4)(k_1 + k_2) \tag{4.8}$$

$$k_4 = f\,(y_n + k_3,\ t_{n+1},\ \bar{x}_{n+1}) \tag{4.9}$$

$$y_{n+1} = y_n + (h/6)(k_1 + k_4) + h/3(k_2 + k_3) \tag{4.10}$$

General Nonlinear Algorithm with Shifted Averaging

The second nonlinear algorithm is based on the preceding method. In this technique, the fast subsystem averages $\bar{x}_a$ and $\bar{x}_b$ are computed over the intervals $\left[t_{n+1/4},\ t_{n+3/4}\right]$ and $\left[t_{n+3/4},\ t_{n+5/4}\right]$ respectively. This difference results in evaluating the fast subsystem variables at quarter steps. The fast system is integrated over two quarter steps with averages computed in the second of those steps. At this point, an approximation to the slow system is computed as shown

$$k_1 = f\,(y_n,\ t_n,\ x_n) \tag{4.11}$$

$$K_2 = f\,(y_n + (h/2)K_1,\ t_{n+1/2},\ x_{n+1/2}) \tag{4.12}$$

$$y^a_{n+1/2} = y_n + (h/4)(K_1 + K_2) \tag{4.13}$$

This value of $y^a_{n+1/2}$ is then used in the fast system integration over the third quarter where the computation of $\bar{x}_a$ is completed. This average is used as in the previous algorithm (equation 4.5 to 4.8) to yield the half step solution of the slow subsystem. The fast subsystem is integrated again over the third quarter step using the new value of $y_{n+1/2}$. The fourth quarter step integration is performed for the fast again with averages compiled. A full step approximation to the slow system is then obtained as follows:

$$K_4 = f\,(y_n + k_3,\ t_{n+1},\ x_{\ n+1}) \tag{4.14}$$

$$y^a_{n+1} = y_n + (h/6)\ (k1 + K4) + (h/3)(k_2 + k_3) \tag{4.15}$$

This approximation value is used in the integration of the fast system from t_{n+1} to $t_{n+5/4}$ to complete the calculation of x_b. With this average, the full step solution is obtained as in the first nonlinear algorithm (equations 4.9 and 4.10).

Programming Conventions

The coding of the partitioned nonlinear algorithms consisted of modifying an existing program (coded by O. A. Palusinski) to include averaging. The original program contains a subroutine INTRGX which updates the next states for the slow nonlinear subsystem according to a four point Runge-Kutta rule. The fast subsystem integration is performed by a Runge-Kutta Merson subroutine called RKM (coded by

J. V. Wait and O. A. Palusinski) which includes provisions for partitioned integration. Among its function is the extrapolation of the slow system variables for the integration of the fast subsytem.

Both partitioned algorithms required a change in INTRGX only. The first partitioned algorithm was completed by adding code to compute averages of the fast variables following returns from subroutine RKM. The second algorithm required a more extensive modification. Here, code was added not only to compute averages, but also to keep track of the shifted time interval. Flow charts for subroutines INTRGX for both algorithms are shown in Figures 4 and 5, and source listings of RKM and the two INTRGX routines may be found in Appendix A.

Use of Nonlinear Algorithms: An Example

The use of the nonlinear algorithms to perform a simulation consists of writing the differential equations for the fast system in $D1 and the differential equations for the slow system in $D2 in accordance with the rules of DAREP (Korn and Wait, 1977, pp. 79-105). In addition, each derivative block must contain a procedural section which links variables needed in that block to the appropriate extrapolated or averaged values (see example in Appendix A). This linking is performed by convention by two user supplied subroutines. LINKW links extrapolated values of the slow system variables to the fast system equations. This is seen in the example in Appendix A which shows that the extrapolated values of the first and sixth state variables of the slow system are needed by the fast subsystem. By convention, LINKW links the average values of the fast system to the slow system. In the example, it is

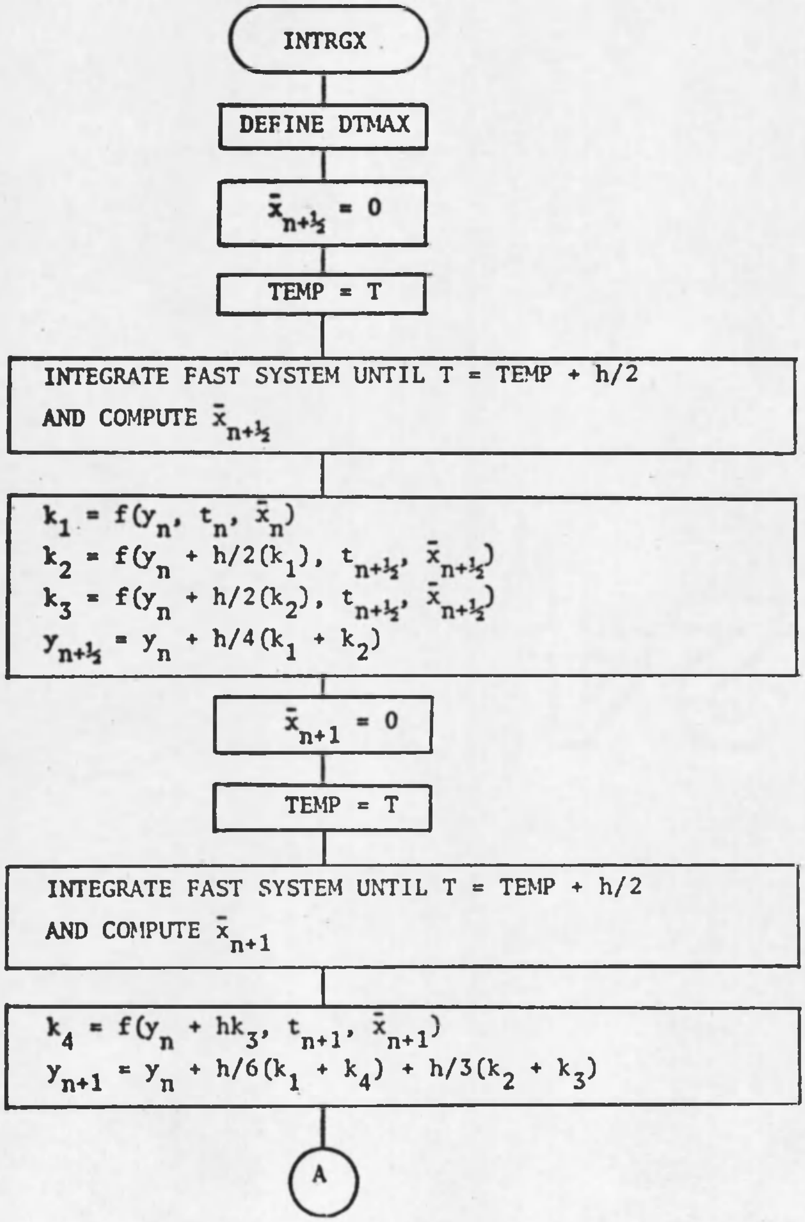

Figure 4. Flow chart of general nonlinear algorithm with averaging

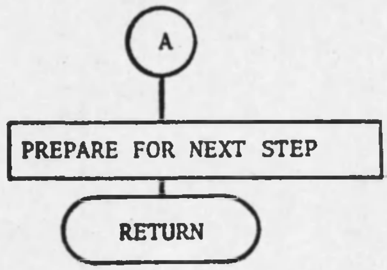

Figure 4--Continued

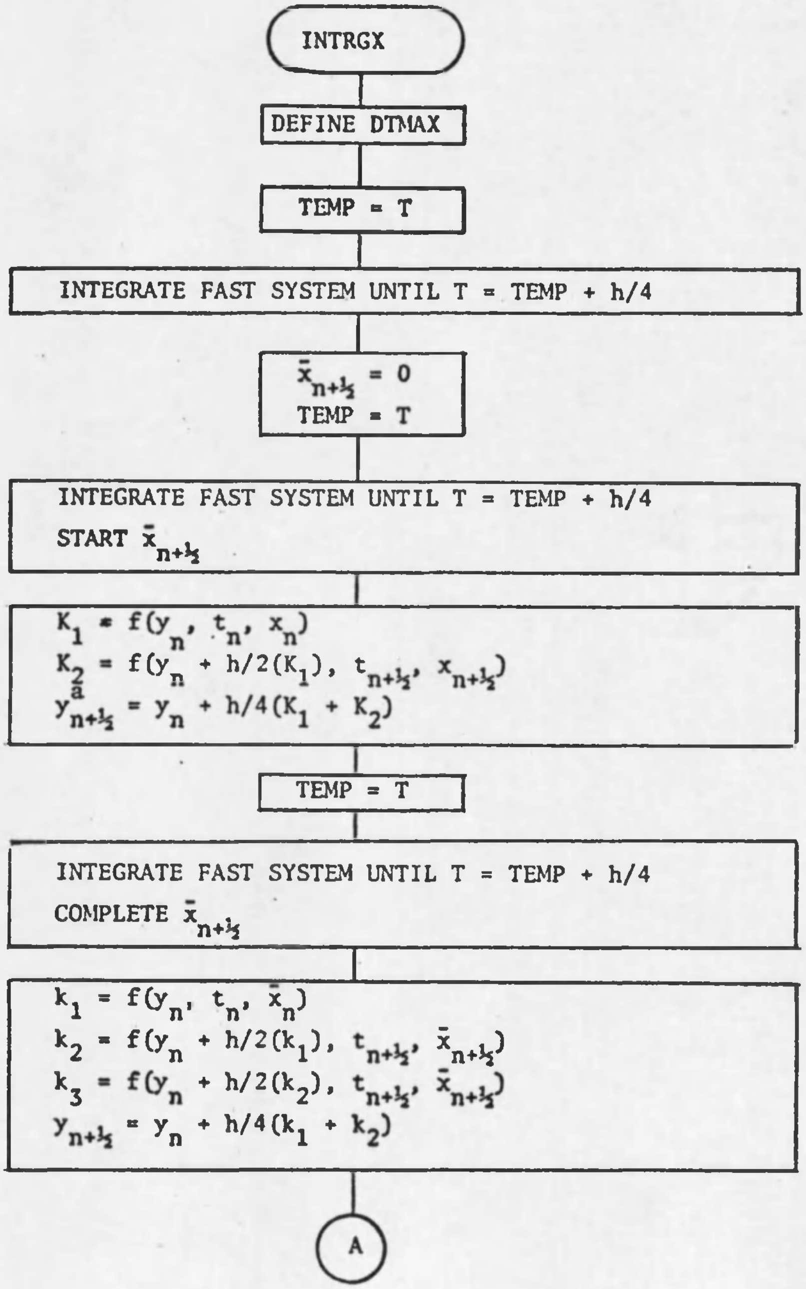

Figure 5. Flow chart of subroutine INTRGX of general nonlinear algorithm with shifted averaging

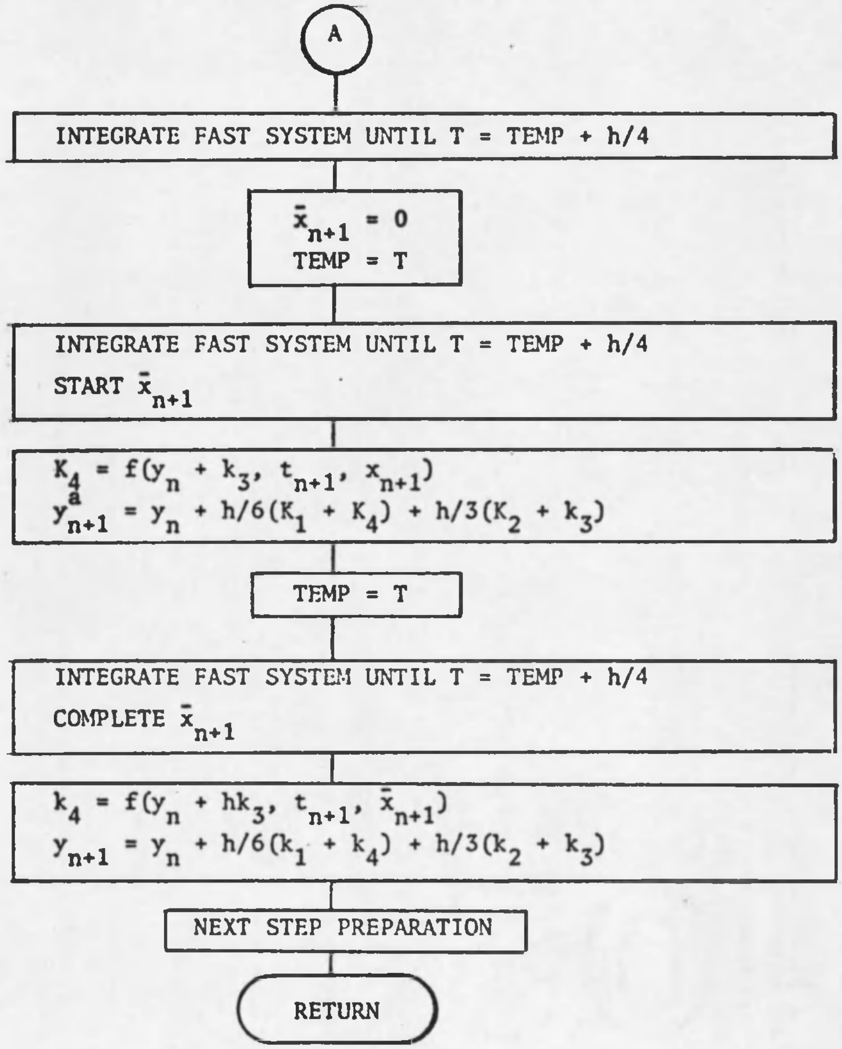

Figure 5--Continued

seen that the average of the fourth state variable of the fast system is needed by the slow subsystem.

Finally, parameter values and initial conditions on both subsystems and output requests are entered.

CHAPTER 5

DESCRIPTION OF EXPERIMENTS

The simulations conducted to test the four algorithms are presented here together with techniques for error estimation. This chapter is divided according to the combined linear-nonlinear experiments and the partitioned nonlinear tests followed by a discussion on errors.

Combined Linear Nonlinear Experiments

Three test examples were used to study these methods. These were a harmonic oscillator, simulation of a servo-controlled pendulum and simulation of a mine-shaft elevator. All three were used with the Improved Euler method. Simulations of the servo-controlled pendulum also were carried out with the Modified Euler method.

The harmonic oscillator problem was used mainly to test the integration algorithm for proper behavior on a simple problem of known solution. The problem consisted of solving the differential equation

$$\ddot{y} + y = 0 \qquad (5.1)$$

This second order equation was broken down into two first order equations:

$$\dot{y} = z \qquad (5.2)$$

$$\dot{z} + y = 0 \qquad (5.3)$$

By setting the initial value of y and $\dot{y}$ to 0 and 100 respectively, a solution of 100 sin t and 100 cos t is obtained for y and $\dot{y}$.

The description of this problem in a format acceptable by the combined linear-nonlinear integration schemes required that equation (5.2) be treated as the nonlinear equation

$$\dot{Y} = WAV \tag{5.4}$$

The linear system was therefore described by

$$\dot{X} = 0 * X + 1 * U \tag{5.5}$$

$$W = 1 * X + 0 * U \tag{5.6}$$

In addition, the coupling equation for U was replaced

$$U = Y \tag{5.7}$$

It should be noted that X, Y, U, and WAV are scalars. Finally, F, defined as

$$F = \frac{\partial (Y, t, \bar{w})}{\partial \bar{w}}$$

in also a scalar (F = 1). (5.8)

The servo-controlled pendulum and mine-shaft elevator models are described in Palusinski and Wait (1978, pp. 14-21) and for the sake of brevity will not be discussed here.

General Nonlinear Experiments

The two nonlinear algorithms were tested first on two partitions of an electric oscillator labelled A and B and then on two partitions of an autopilot system named P1 and P2. The two partitions of the electronic oscillator are discussed in Palusinski (1977b, pp. 9-15). The general block diagram of the autopilot system is shown in Figure 6.

The autopilot system model was developed by modeling each of the subsystems shown in the block diagram and then combining all these subsystems to produce the overall model.

The first subsystem dealt with was the vertical sensing unit which may be thought of as a gyroscope which converts angular deflection to a voltage. This gyro may be described by the transfer function (Gille, Pelegrin, and Decaulne, 1959, pp. 710-712).

$$\frac{u(s)}{\emptyset(s)} = k_d \frac{s}{1 + (1.2/40)s + s^2/1600} \tag{5.9}$$

where $\emptyset$ is the pitch angle input to the gyroscope, u is the voltage output and k_d is a constant. This results in the following differential equation

$$\ddot{u} = -48\,\dot{u} - 1600\,u + 1600\,k_d\,\dot{\emptyset} \tag{5.10}$$

with zero initial conditions.

The next system considered was the compensating network and amplifier, and is defined by the following transfer functions (Gille et al., 1959, pp. 713-716).

$$\frac{W(s)}{E(s)} = \frac{k_a}{1 + 0.10s} \tag{5.11}$$

$$\frac{X(s)}{W(s)} = \frac{1}{10} \quad \frac{1 + 10\tau s}{1 + \tau s} \tag{5.12}$$

where E is the input to this subsystem, W is the output of the amplifier, K_a is the gain of the amplifier, X is the output and τ is the time constant of the compensating network. These result in the differential equations.

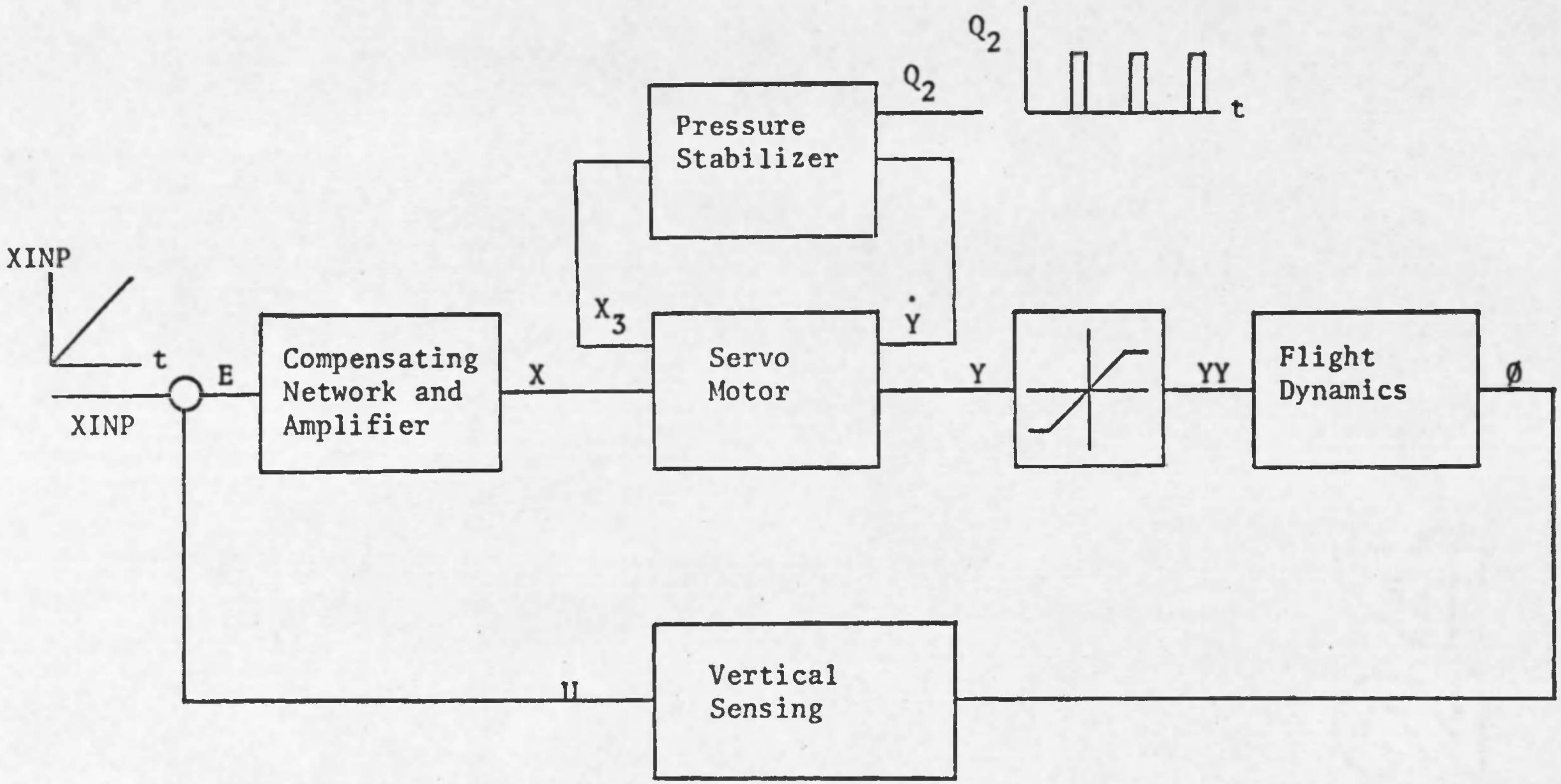

Figure 6. Block diagram of autopilot system

$$\ddot{w} = -100\ W + 100\ K_a\ E \tag{5.13}$$

$$\dot{x} = -X/ \quad + K_a W/(10\ \tau \quad) + 10\ K_a\ \dot{W}/\tau \tag{5.14}$$

with initial conditions.

The third block consisting of flight dynamics was taken from Korn and Korn (1956, pp. 115-124) and the differential equations describing the model are shown below.

$$\dot{V} = -.22V - 16.6C - g\cos\theta,\ \sin\theta \tag{5.15}$$

$$\dot{O} = (0.237\ V + 238\ C - 26.6\ AZ\ (YY)$$

$$+ 1.68\ \dot{\emptyset} + g\sin\theta,\ \cos\theta) \tag{5.16}$$

$$\ddot{\emptyset} = M_V V - 11.9\ C + 10.3\ AS\ (YY) - 679\ \dot{\emptyset} \tag{5.17}$$

$$c = \emptyset - \theta \tag{5.18}$$

where V is the velocity, θ is the lift angle, ∅ is the pitch angle of the plane. The function AS(YY) is described by way of a table containing a set of points approximating

$$AS\ (YY) = \sin^{-1}\ (YY/K) \tag{5.19}$$

where K was chosen to produce $-0.5 < \text{rad}\ AS(YY) < 0.5$ rad. Again, zero initial conditions are assumed.

Finally, the hydraulic servomotor and pressure stabilizer were investigated. The hydraulic servo may be determined by

$$\ddot{Y} = -R_m\ \dot{Y}/J + k\ S_o PX/J \tag{5.20}$$

$$P = X_3 . 10^5 \tag{5.21}$$

where Y is the displacement output of the servo, P is the pressure obtained from the pressure stabilizer through X_3, R_m is a damping factor, So is the surface area of the actuator piston, and J is the moment of inertia of the piston.

The pressure stabilizer is presented in Palusinski, Skowronek, and Znamirowski (1976, pp. 211-218) and the equations are repeated here

$$\dot{X}_1 = (8\ X_2)\ 250 \tag{5.22}$$

$$\dot{X}_2 = (-6.59\ X_2 - 0.0146\ F(X_1)\ X_3 - 0.54X_1 + 28.7X_3 - 655.227)\ 250 \tag{5.23}$$

$$\dot{X}_3 = (78.674 - 0.638\ F(X_1)\sqrt{X_3} - 0.67X_2 + 2.78\ Q_1 + Q_2)\ 250 \tag{5.24}$$

$$Q_1 = K_p\ \dot{Y} \tag{5.25}$$

where X_1 is a valve displacement, X_2 is proportional to the speed of valve movement, X_3 is the pressure in atmospheres, and Q_2 is a periodic train of pulses representing disturbance in the system. The function $F(X_1)$ described in Palusinski et al. (1976, p. 215) was approximated by

$$F(X_1) = 0.25\ X_1 \tag{5.26}$$

In the first partition investigated (P1), the hydraulic servo and the pressure stabilizer were placed in the fast subsystem and the vertical sensing unit, compensating network, and flight dynamics comprised the slow subsystem. In the second partition (P2), the fast subsystem was made up of only the pressure stabilizer.

Error Computation

All simulations were performed by executing the algorithms with the appropriate differential equations in conjunction with DAREP of the Control Data Corporation model CYBER 175 computer at The University of Arizona. Each of the simulations was first run using a Runge-Kutta Merson rule with no partitioning of the system. When a desired response was obtained, several more runs using the same Runge-Kutta Merson rule were performed each with a smaller error bound than the previous run. When two consecutive runs were found to be identical to six significant digits, the latter of the two was taken as the final solution. The maxima of the variables were noted in each case and scaled to a value of 100. These scaled values were stored on a permanent file and comprised the benchmark or reference values for error computation. Each experiment was run for various values of h, various partitions, and characteristic parameters and the same variables were scaled by the same scale factors. The differences between the values obtained in these runs and those obtained in the benchmark thus constitutes the percent error in those variables. For each experiment, the peak percent errors were compiled in the form of tables (Appendix C). Similar tables were prepared for the combined linear-nonlinear tests not using averaging by O. A. Palusinski. The method used in these tests has been labelled CRK2HI.C (Palusinski, 1977b) standing for Combined Runge-Kutta method based on the Improved Euler Method and shall be referred to as such from here on. A short description of this method appears in Appendix D. The general nonlinear experiments were run using the original program from which the averaged nonlinear programs were derived to produce tables have all been placed in Appendix C and a comparison between the averaged methods and the methods that do not use averaging is made in Chapter 6.

CHAPTER 6

RESULTS AND CONCLUSIONS

The comparison of the four algorithms employing averaging to similar algorithms that do not use averaging is presented here followed by conclusions that were drawn from the results. The comparison was performed by examining mean peak fractional errors, the variables with the worst peak errors, and execution times for all experiments except the harmonic oscillator.

For each method and experiment, the mean peak error is defined to be the average of all peak fractional errors obtained for a particular step size, and the variable with the worst peak error (Chebyshev measure) for the largest measure step size in that experiment is defined to the the worst-case peak error. The execution time or run time is the time taken by the Central Processing Unit to solve the initial value problem in question using a particular method and step size.

The "cost" of simulation (Palusinski, 1978, p. 38) for a particular step size is defined as

$$\text{COST} = \text{WORST CASE ERROR} + \text{RUN TIME} \qquad (6.1)$$

Results of Combined Experiments

The plot of mean peak errors versus step size obtained for the pendulum problem is shown in Figure 7. The Improved Euler, Modified Eulwe, and CRK2HI.C (without averaging) methods were employed. It is seen that averaging raises mean peak error by as much as three times in the Improved Euler case (h = 0.03) and over ten times in the Modified Euler case (h = 0.09). However, the graph displayed in Figure 8 shows that the worst-case peak error (corresponding to variable TAU1) is improved by as much as 50% (at h = 0.09) using the Improved Euler method, but is still worse by as much as 2.5 times (at h = 0.09) for the Modified Euler algorithm. The execution times for the three algorithms are seen in Figure 9. It is observed that the Improved Euler method shows a speed improvement of up to three times as fast and the Modified Euler method up to twice as fast as CRK2HI.C. In addition, these execution times range from 0.01 to 0.1 seconds compared to a benchmark execution time of 0.787 seconds.

The costs of these methods are shown in Figure 10. As seen, the cost of the Improved Euler method is about a third less than that of CRK2HI.C, but the Modified Euler shows costs three times higher.

The plot of the mean peak errors for the mine shaft experiment is seen in Figure 11. Here, the combined algorithm with averaging is noted to have up to a 50% reduction in mean peak error for small values of h, but for larger h, displays up to twice as much error (H = 0.12). It is also noted that the mean peak error curve is smoother for the

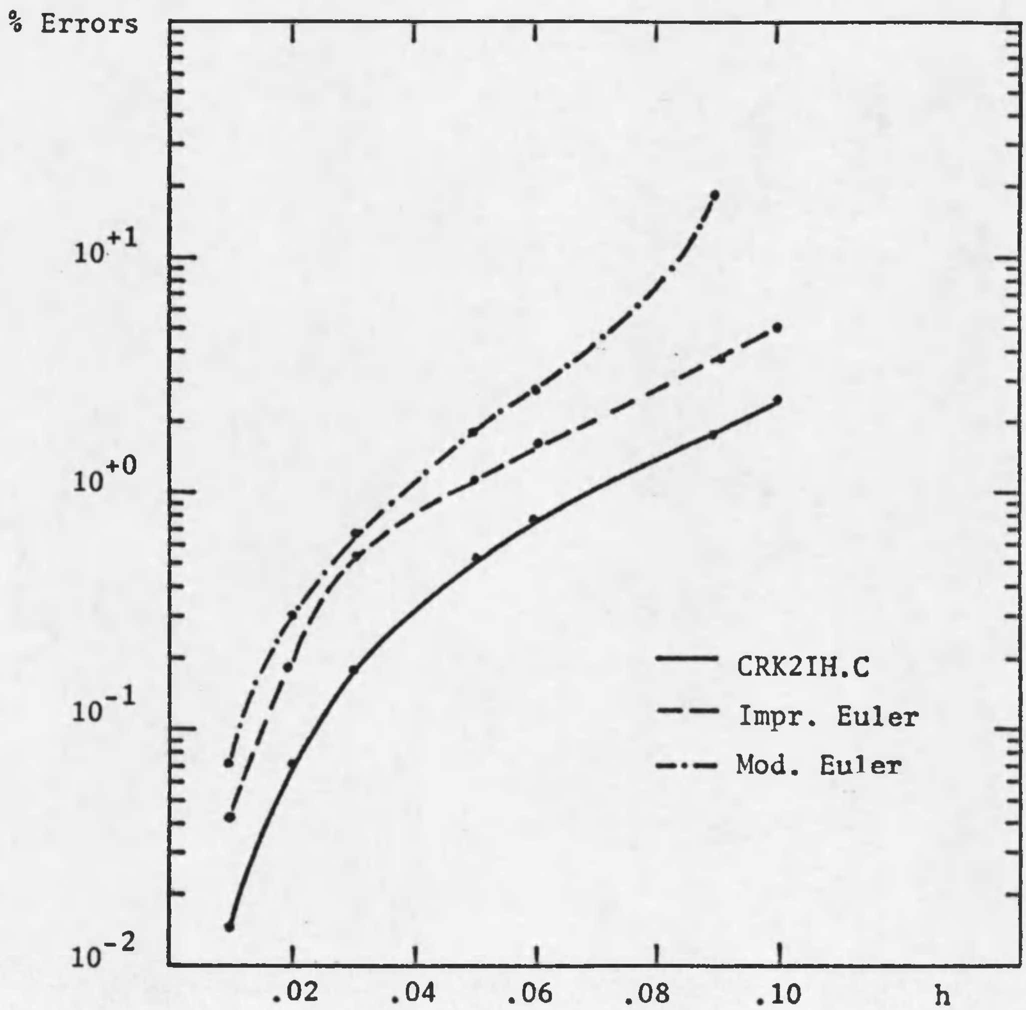

Figure 7. Mean peak error vs. h. Pendulum.

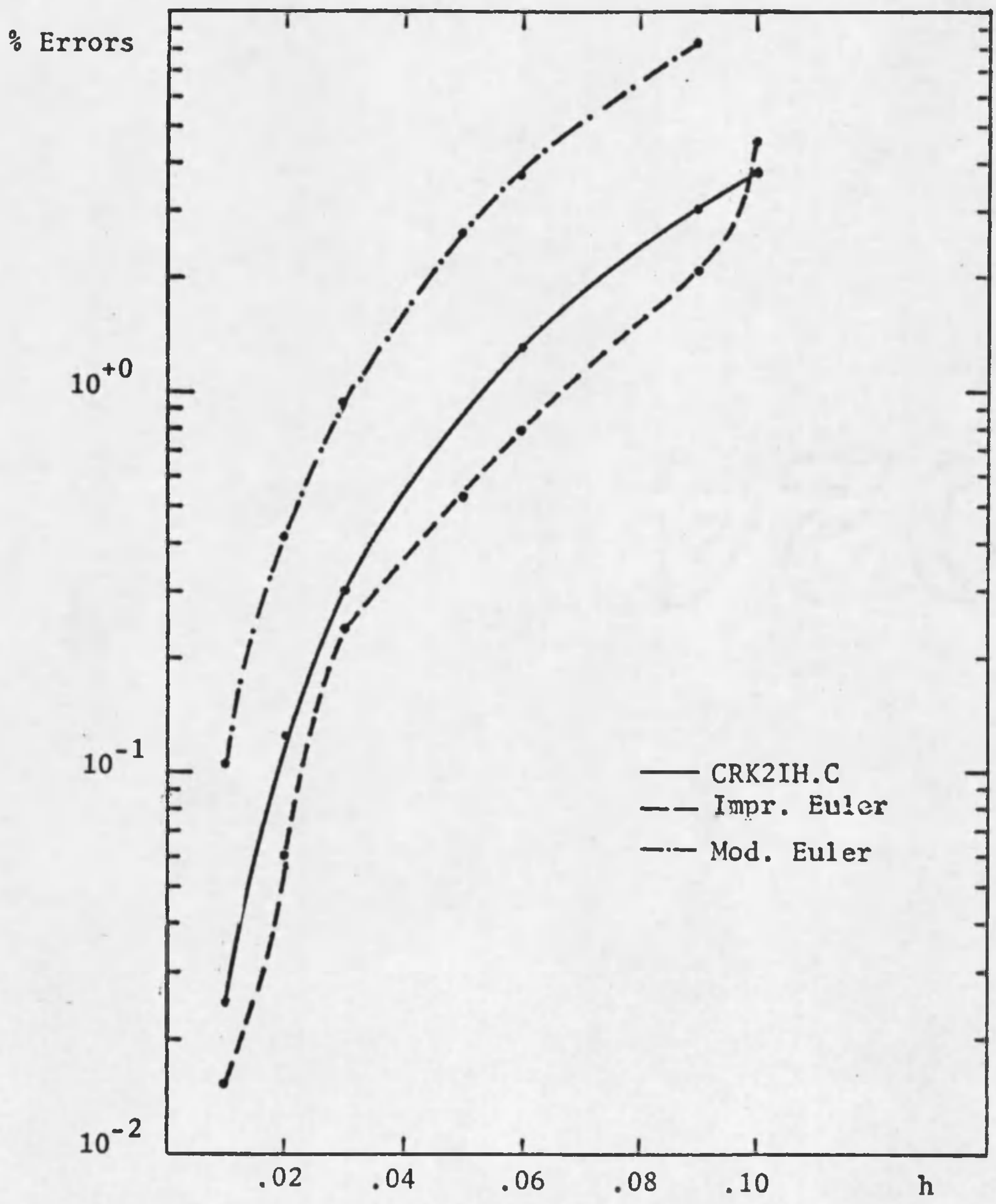

Figure 8. Peak error in TAU1 vs. h. Pendulum

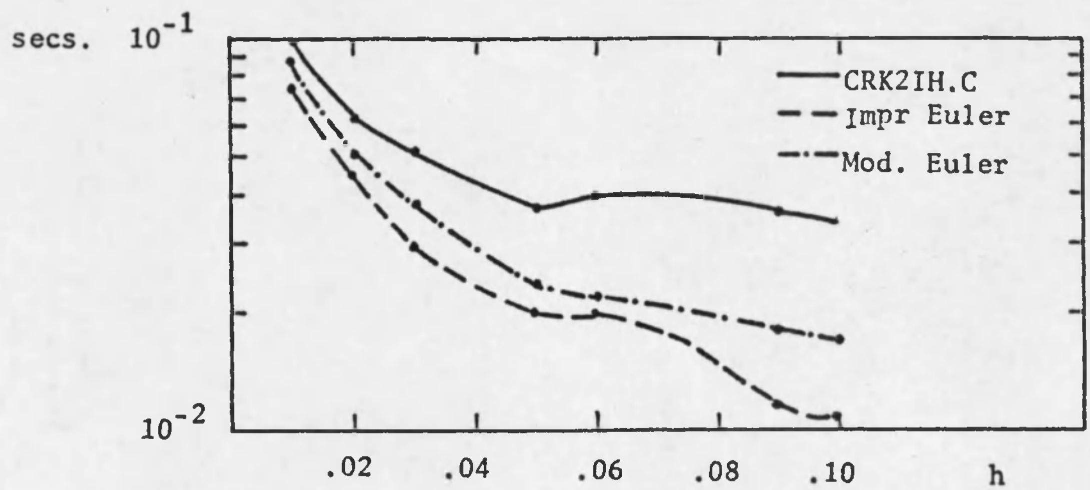

Figure 9. Execution times vs. h. Pendulum

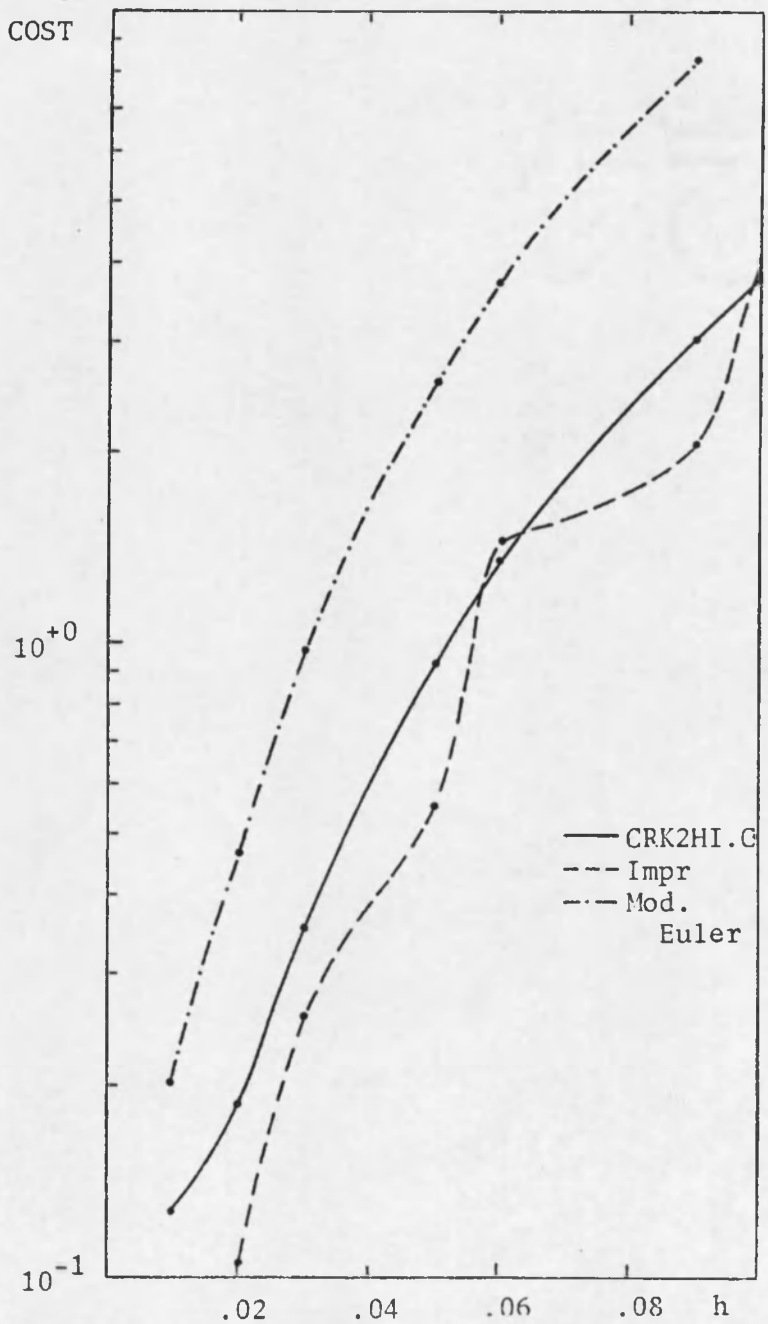

Figure 10. Pendulum Cost

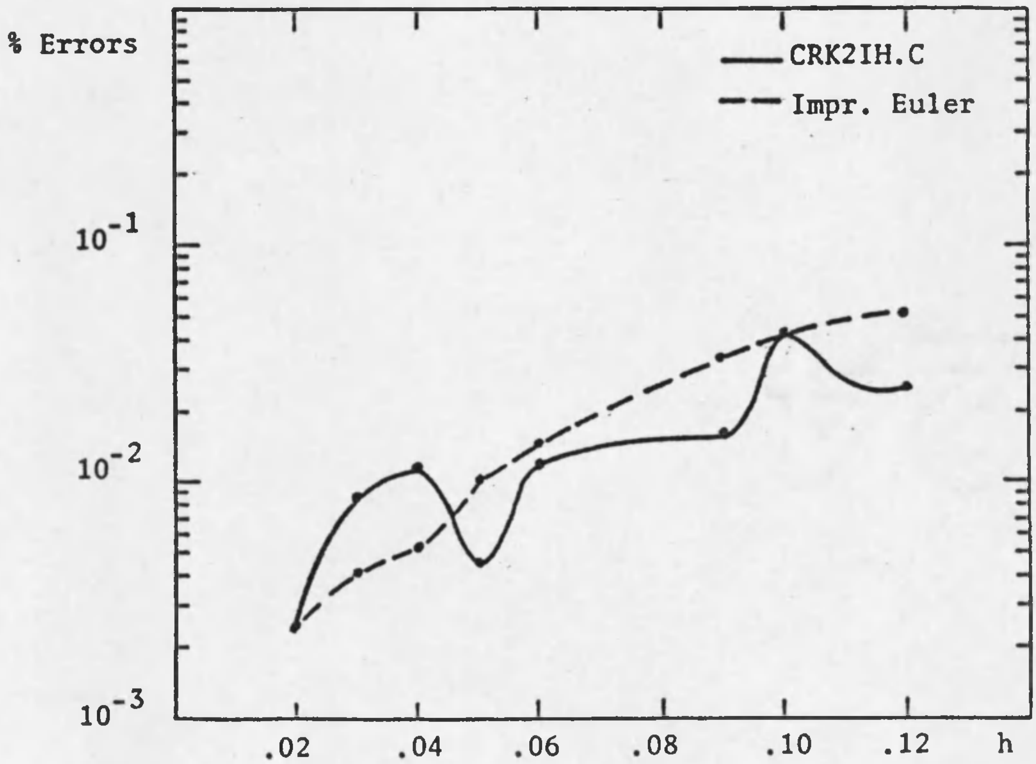

Figure 11. Mean peak error vs. h. Mine-shaft

averaged method. The worst-case peak errors (noted for the variable V) are plotted in Figure 12. Again, no real improvement is noted, but as before, the averaged case results in a smoother curve which appears to be an average of the oscillatory worst-case error curve obtained for CRK2HI.C. The corresponding execution times, ranging from 0.13 to 0.32 seconds, for the two methods are displayed in Figure 13. The averaged method results a speed improvement of approximately 25%. The mine-shaft benchmark took 33.15 seconds to run. The costs of these two methods is shown in Figure 14. As seen, the cost of the averaged method is about 80% that of the method without averaging.

The speed improvement of the combined methods over the CRK2HI.C method may be justified by noting that the latter method is a more complicated version of the Improved Euler without averaging in that extra half-step computations are employed.

Results of General Nonlinear Experiments

The mean peak errors for partition A of the electronic oscillator are plotted in Figure 15. The best mean errors correspond to the algorithm not using averaging followed by those of the shifted averaging algorithm, which are approximately 10 times worse, and those of the nonlinear algorithm with shifted averaging, which are about 100 times worse. These results may be attributed to the fact that the coupling variable that is averaged is itself an average of the oscillatory variables in the fast system. Therefore, any further averaging cannot help. The execution times, ranging from 5.88 to 25.66 seconds, are seen in Figure 16.

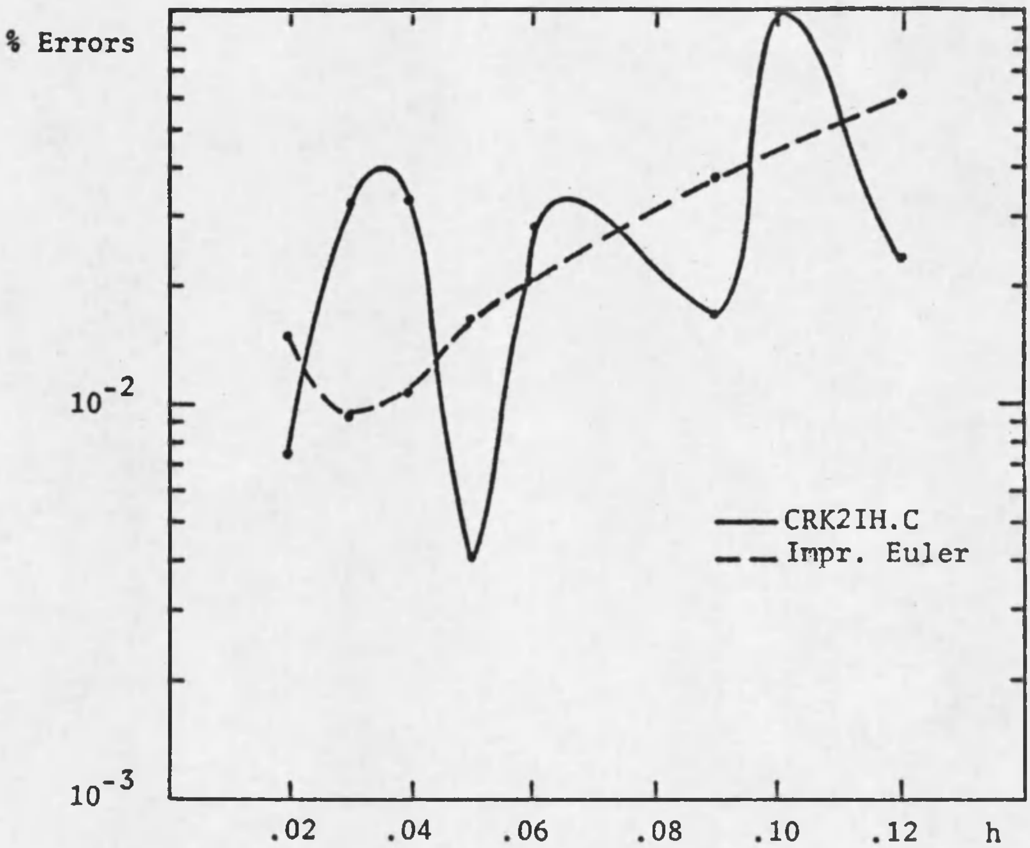

Figure 12. Peak error in V vs. h. Mine-shaft

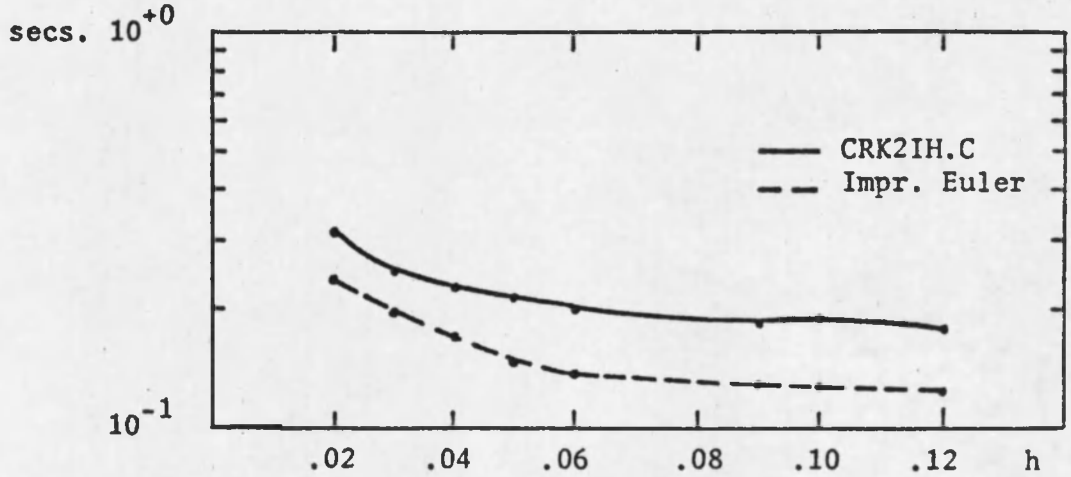

Figure 13. Execution times vs. h. Mine-shaft

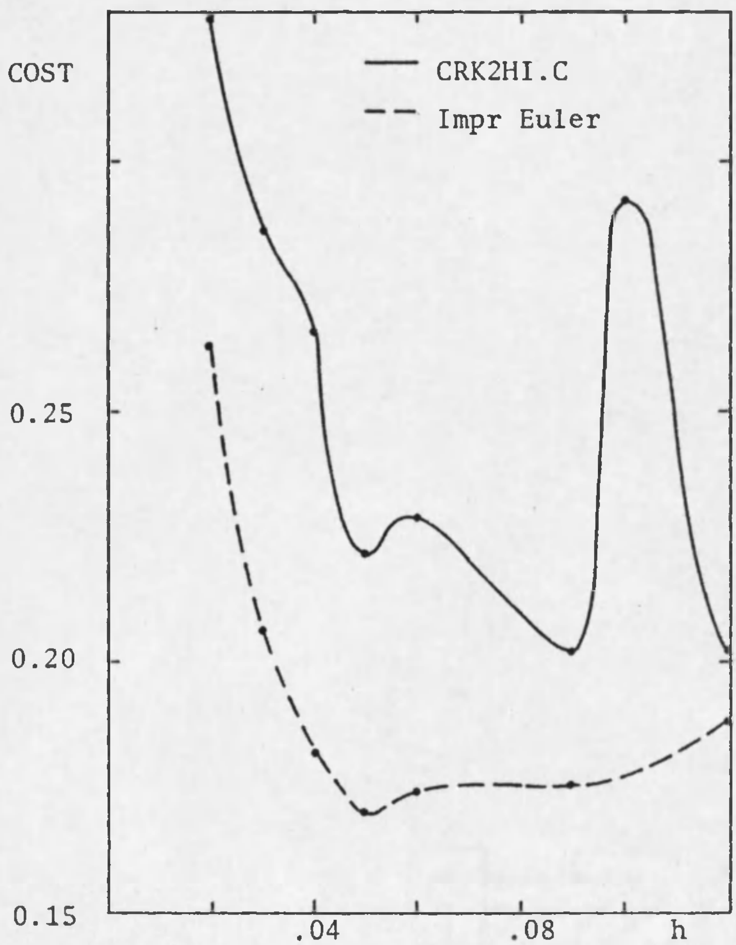

Figure 14. Mine-shaft cost

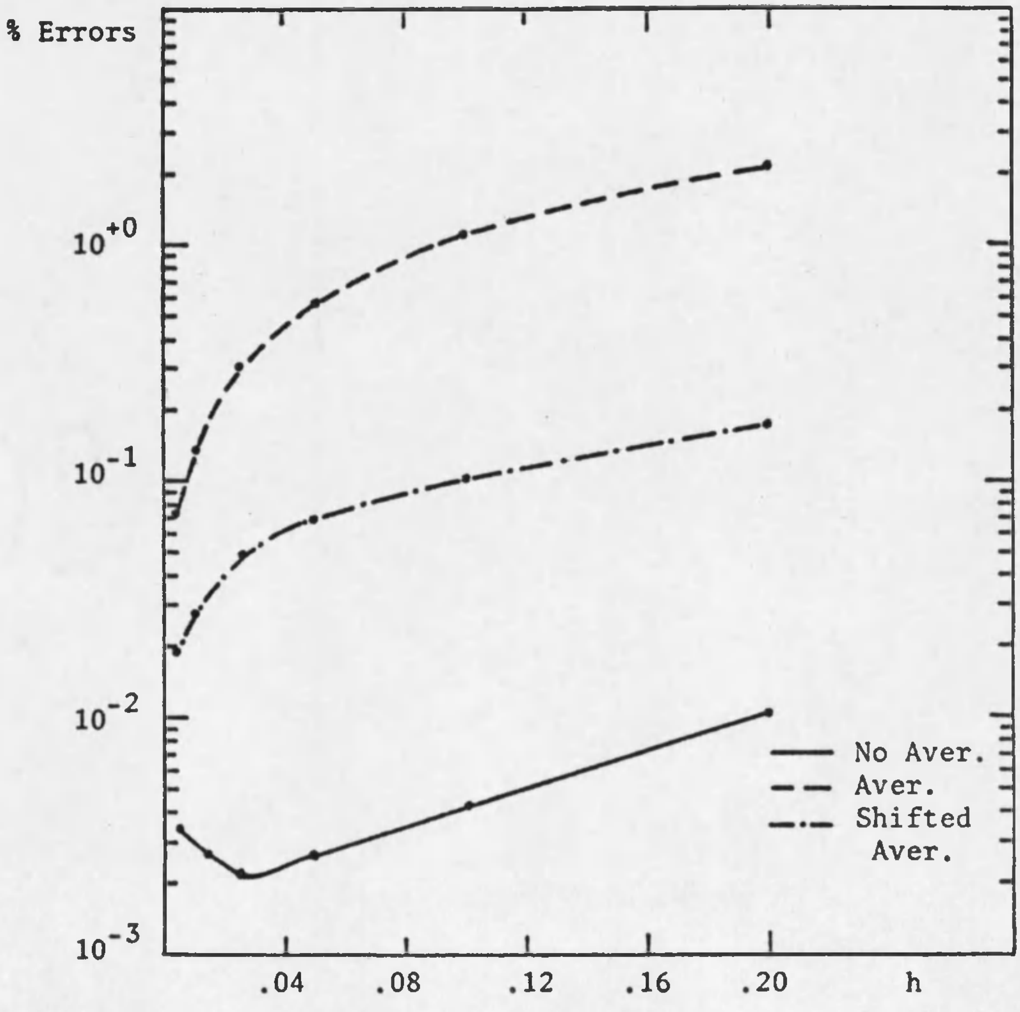

Figure 15. Mean peak error vs. h. Oscillator (partition A)

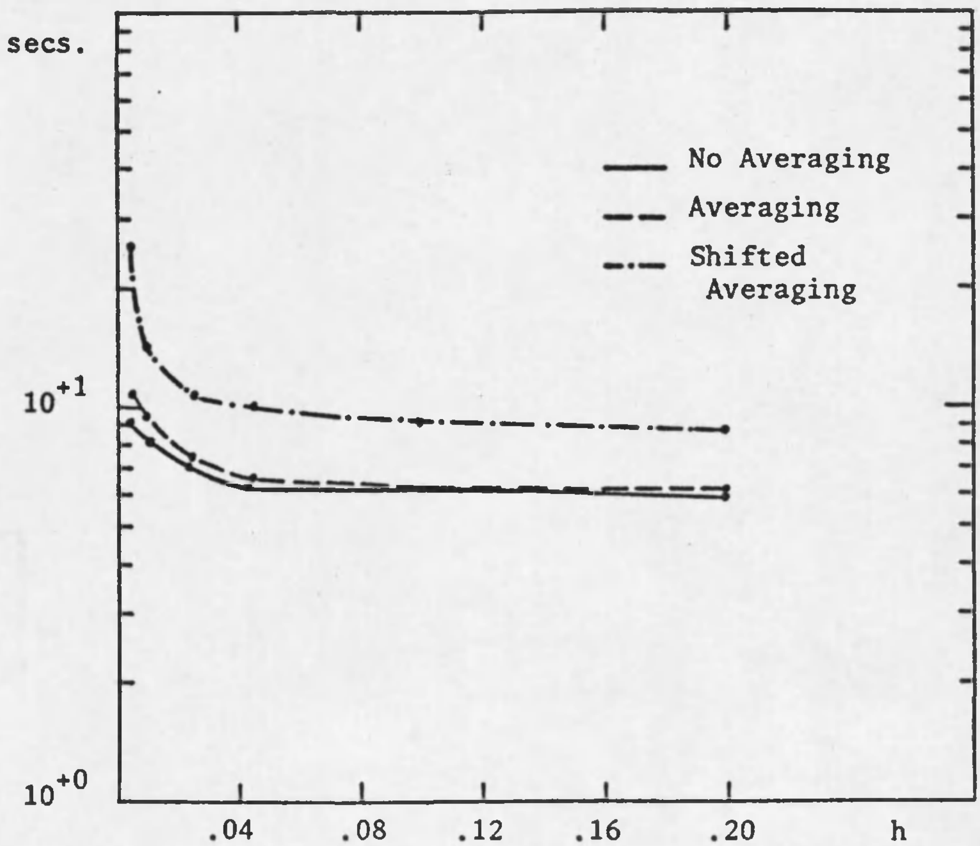

Figure 16. Execution times vs. h. Oscillator (partition A)

As seen, simple averaging slows execution slightly, but shifted averaging reduces speed by as much as twice. The oscillator benchmark required 55.698 seconds to execute.

The mean peak errors for partition B are plotted in Figure 17. Here the coupling variable that is averaged is very fast and averaging improves these errors for the larger step sizes (up to 50% reduction in error for the shifted averaging case and 35% reduction for simple averaging).. The plot of worst-case peak errors (corresponding to X_2) for the same partition as seen in Figure 18 shows improvements of comparable magnitudes for the larger step sizes when averaging is used. The execution times for partition B are very close to those obtained for partition A so that Figure 16 may be considered as an estimate of speed performance for partition B. The costs for these methods for the simulation of partition B are displayed in Figure 19. Simple averaging raises the cost by approximately 10% for smaller values of h, but lowers the cost by almost 20% at h=0.2. Shifted averaging costs from 10% to 150% higher than the method without averaging.

The mean peak errors for partition 1 of the autopilot model were identical for the nonlinear algorithms with simple averaging and without averaging. The shifted averaging mean errors were significantly larger. In order to obtain a meaningful comparison, only those variables, namely Ø, X, and Y, which were noted to differ in the two identical cases, but whose magnitudes were so small that the differences were not affecting the original mean errors, were considered in a recomputed mean peak errors. This plot of the recomputed mean errors for the simple averaged

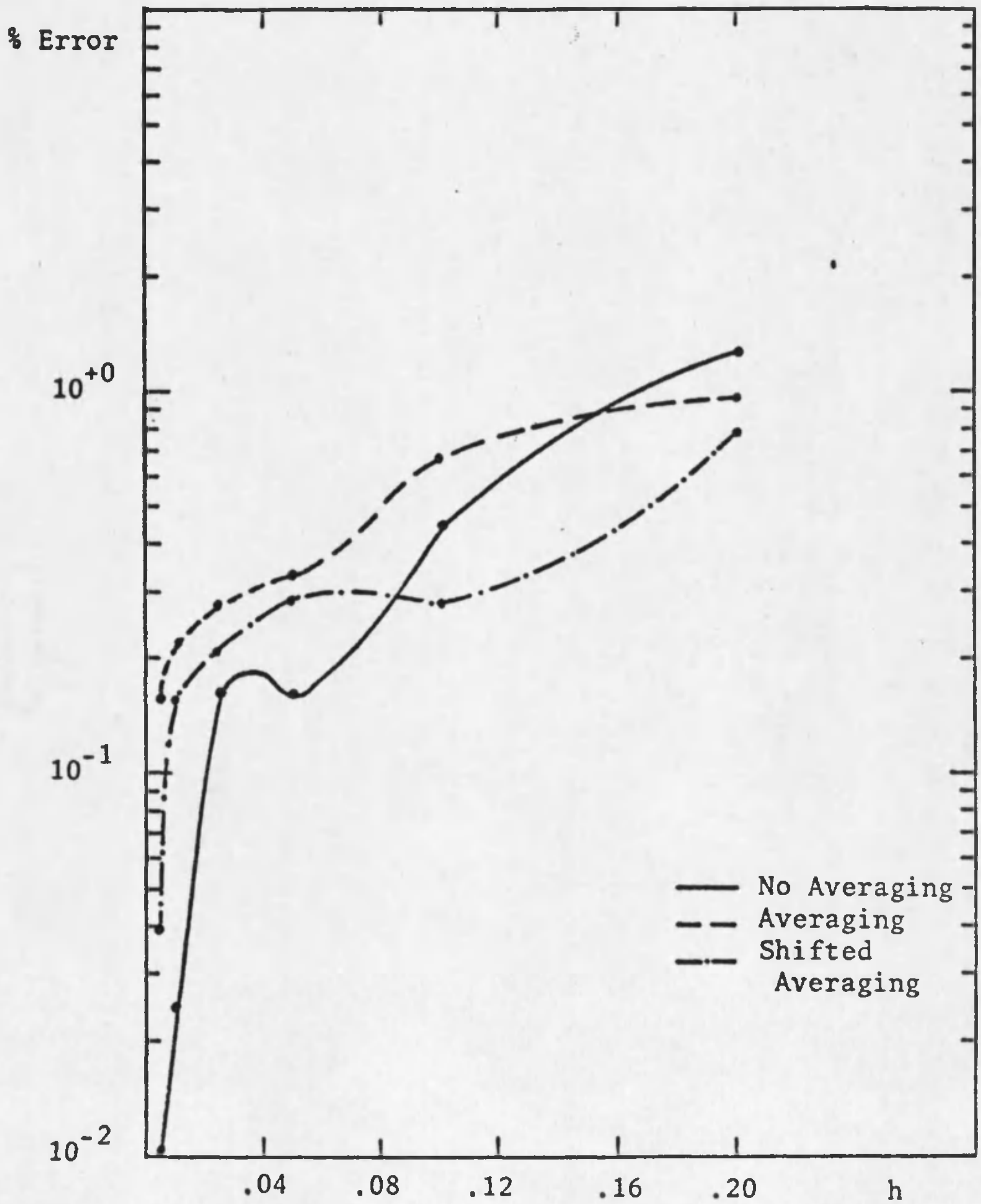

Figure 17. Mean peak errors vs. h. Oscillator (partition B)

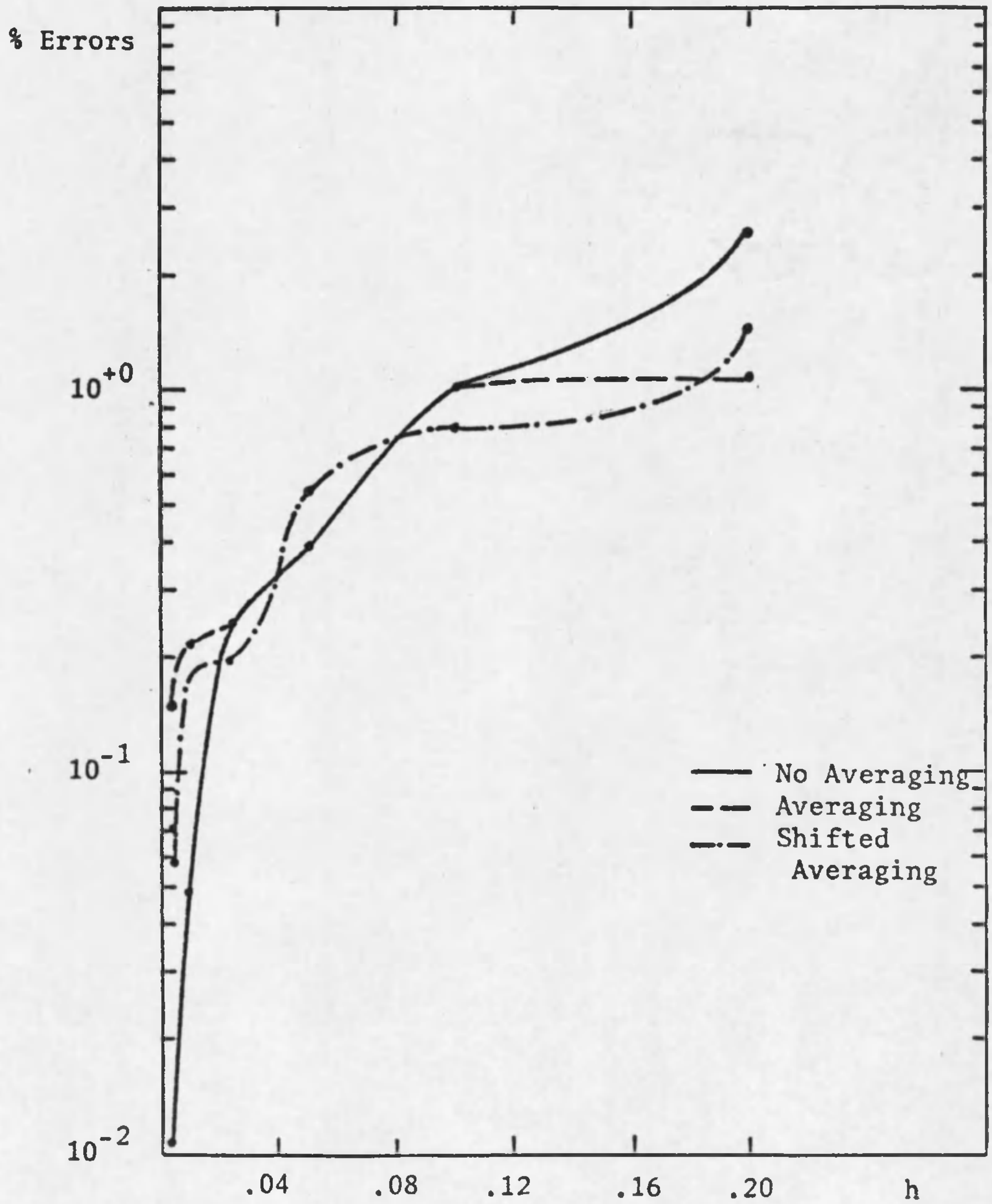

Figure 18. Peak error in X_2 vs. h. Oscillator (partition B)

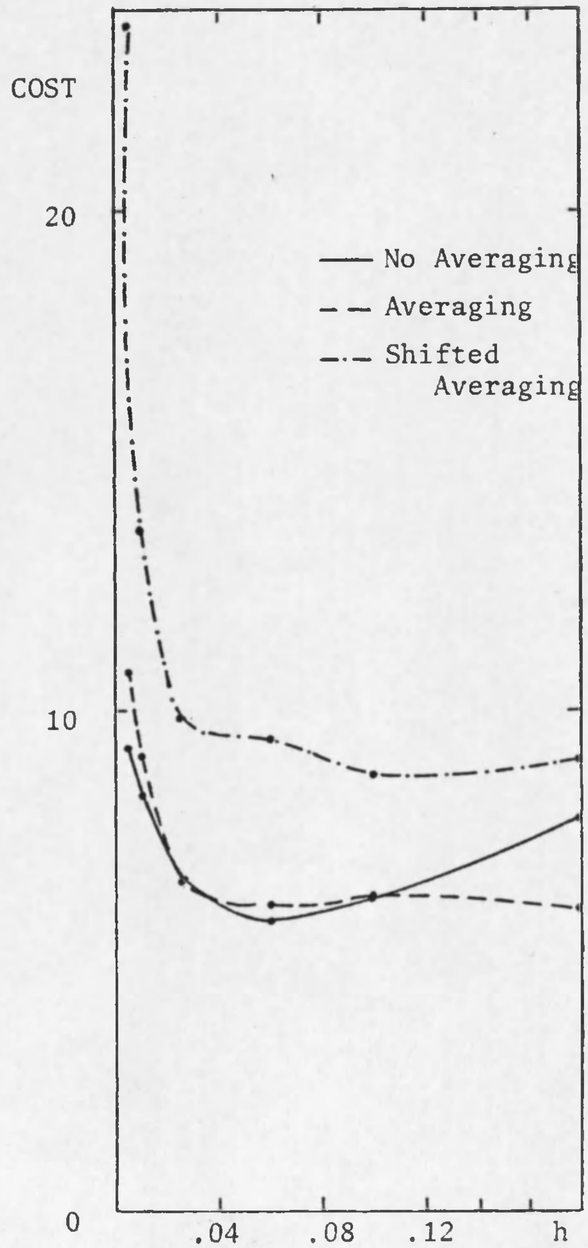

Figure 19. Oscillator (partition B) cost

and the averaged case is shown in Figure 20. Here, differences are only slight, but the plot of worst-case errors, displayed in Figure 21, show that the averaging method results in up to a 50% improvement. Execution times ranging from 2.91 to 70.25 seconds for partition 1 are plotted in Figure 22. Again the simple averaged method is only slightly slower than the method without averaging, but the shifted averaging method is slightly 3 times slower than both. The benchmark ran in 71.74 seconds. The costs for these methods for the simulation of partition 1, seen in Figure 23, again indicate slightly higher costs for the simple averaging method with costs for the shifted averaging case ranging over twice that of the non averaged case. Simple averaging improves the mean peak errors and worst case errors (again corresponding to X_2) slightly as seen in Figures 24 and 25, but shifted averaging is still worse by as much as a factor of 2. The execution times for the second partition are also very close to those of partition 1, and so the run times of Figure 22 may serve as an indicator for the performance of partition 2.

Conclusions

The results obtained from this study have shown that two of the averaging methods considered are useful in the simulation of partitioned systems. As for the other methods, it was observed that existing methods that do not use averaging, in general, yield improved errors and lower execution times. This is especially true of the Modified Euler method where percent errors were as much as ten times larger than the

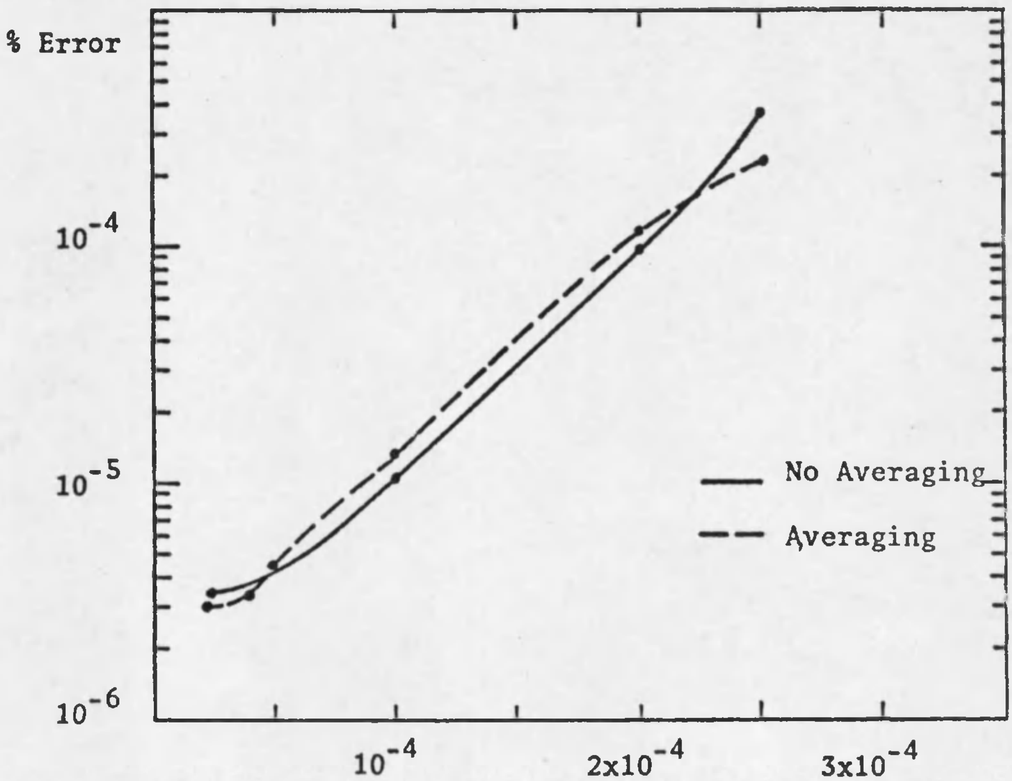

Figure 20. Mean peak error vs. h. Autopilot (partition 1)

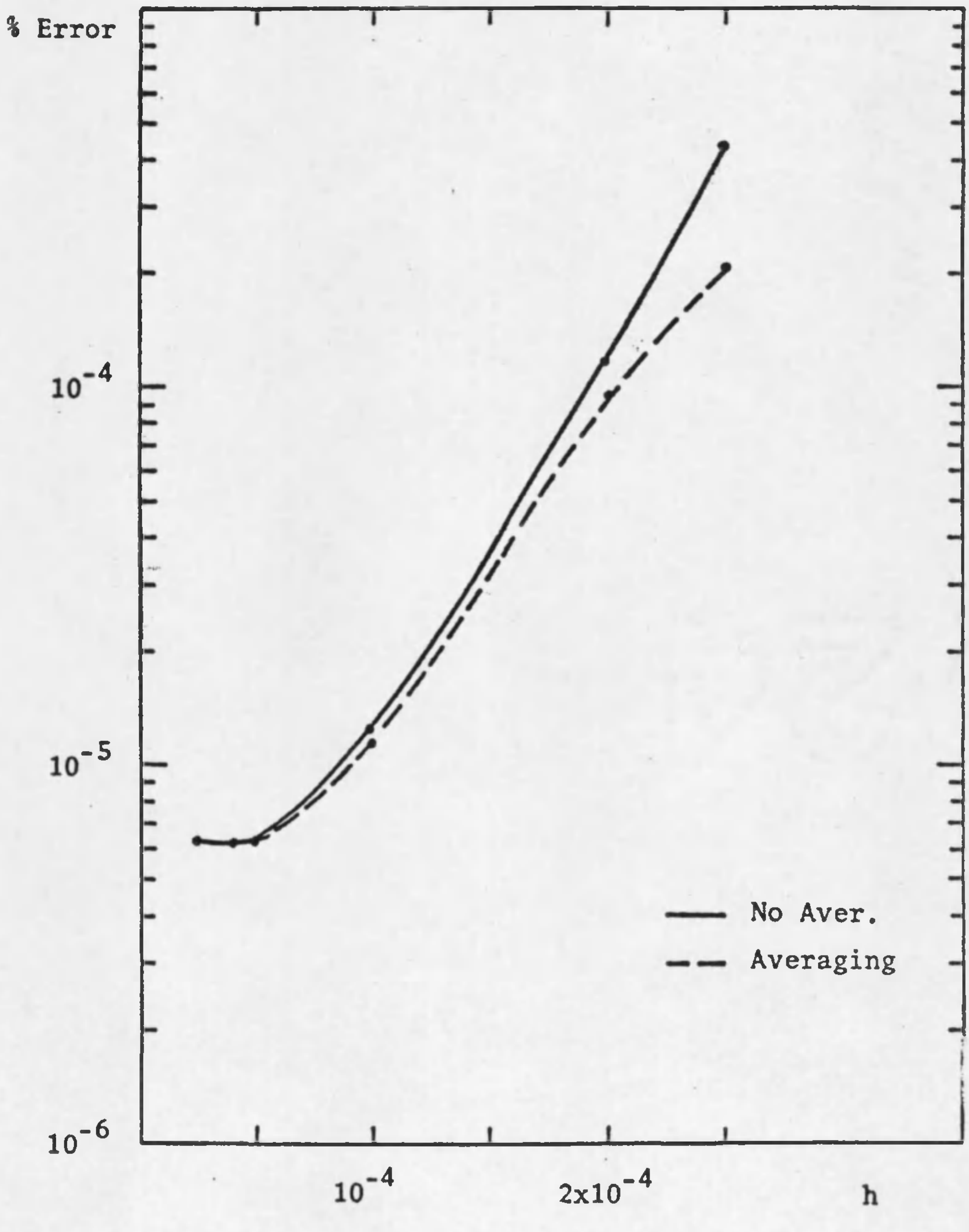

Figure 21. Peak error in X_2 vs. h. Autopilot (partition 1)

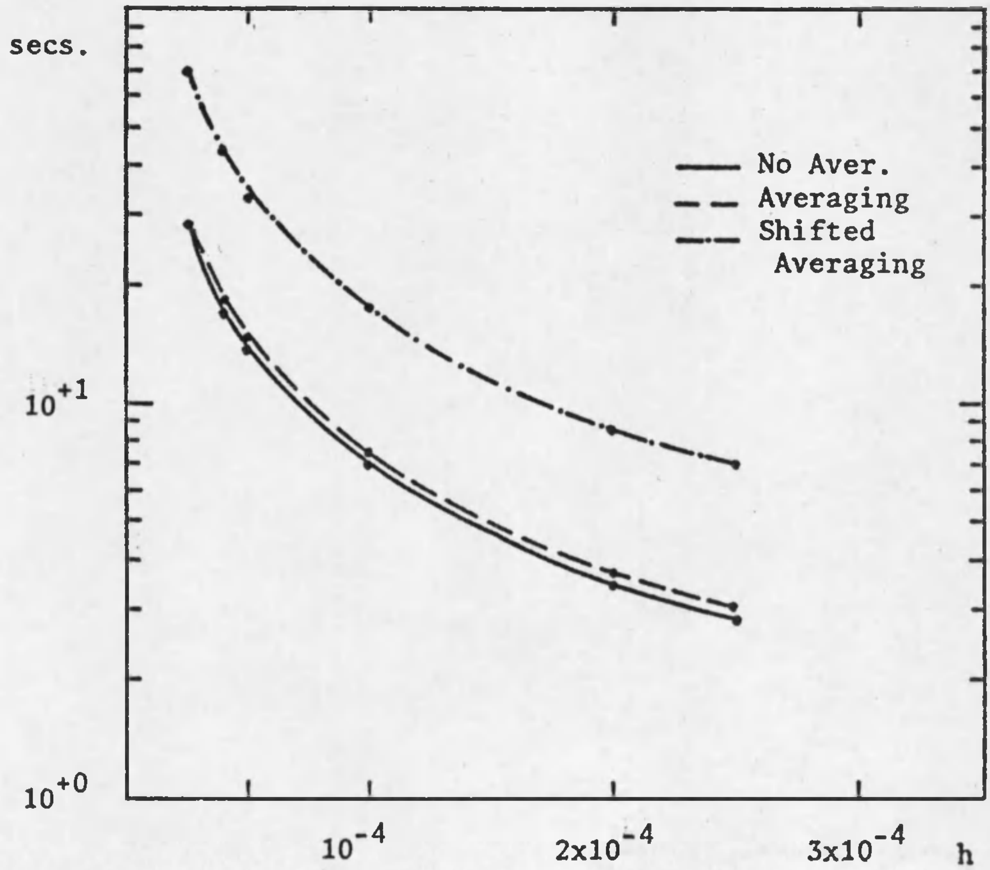

Figure 22. Execution times vs. h. Autopilot (partition 1)

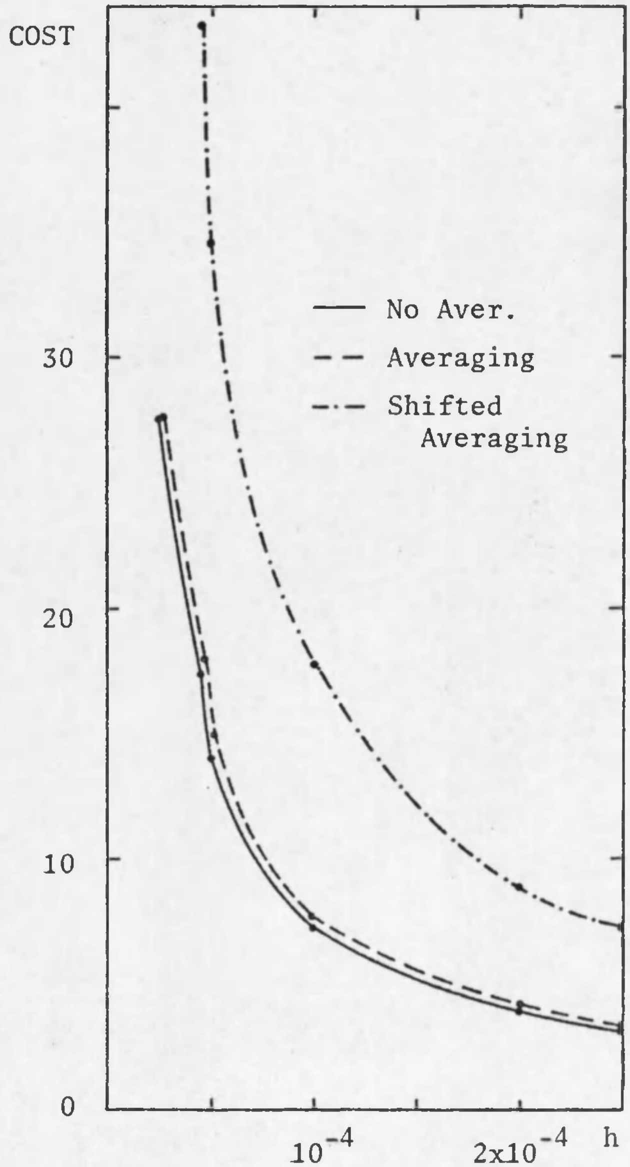

Figure 23. Autopilot (partition 1) cost

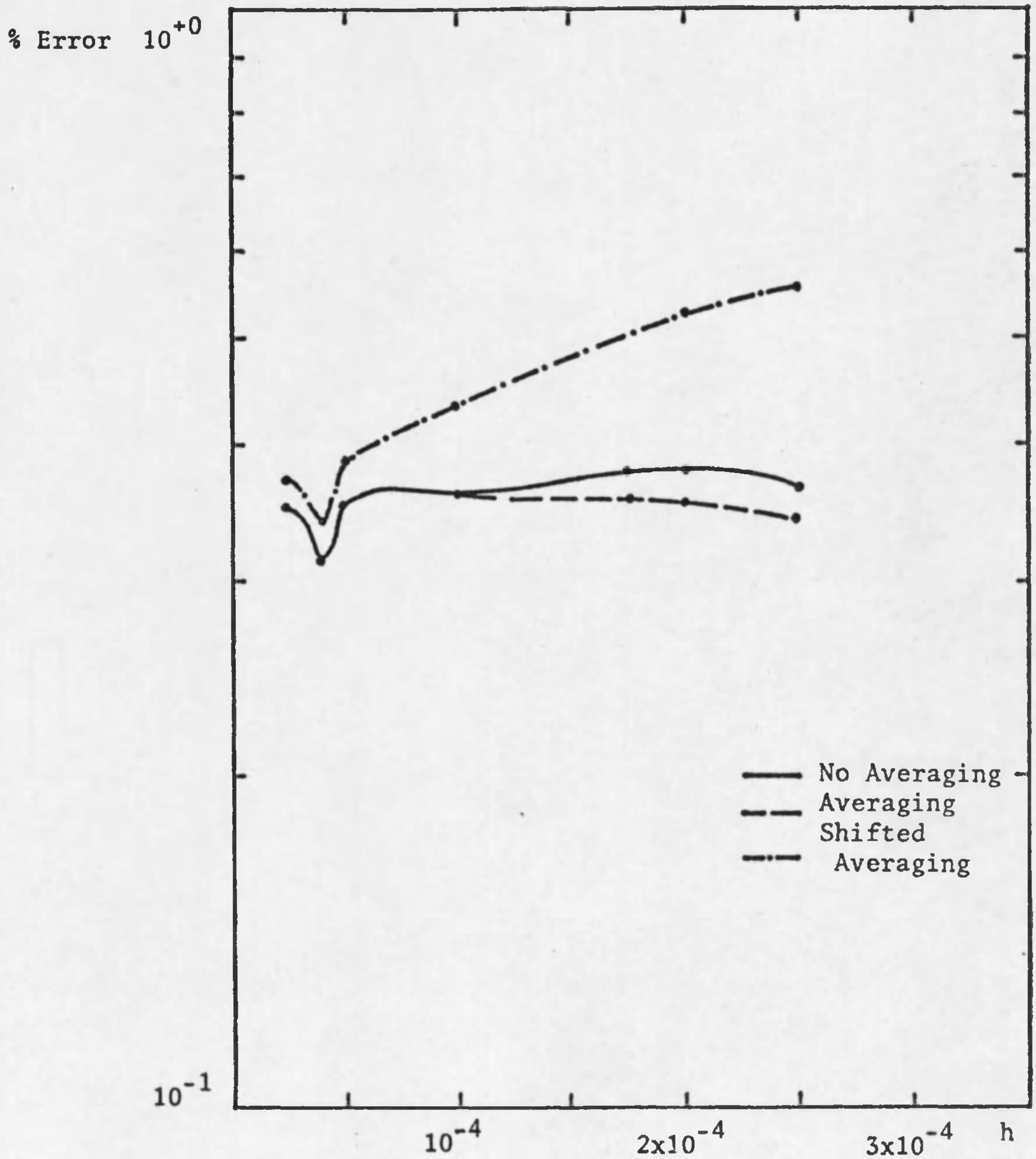

Figure 24. Mean peak error vs. h. Autopilot (partition 2)

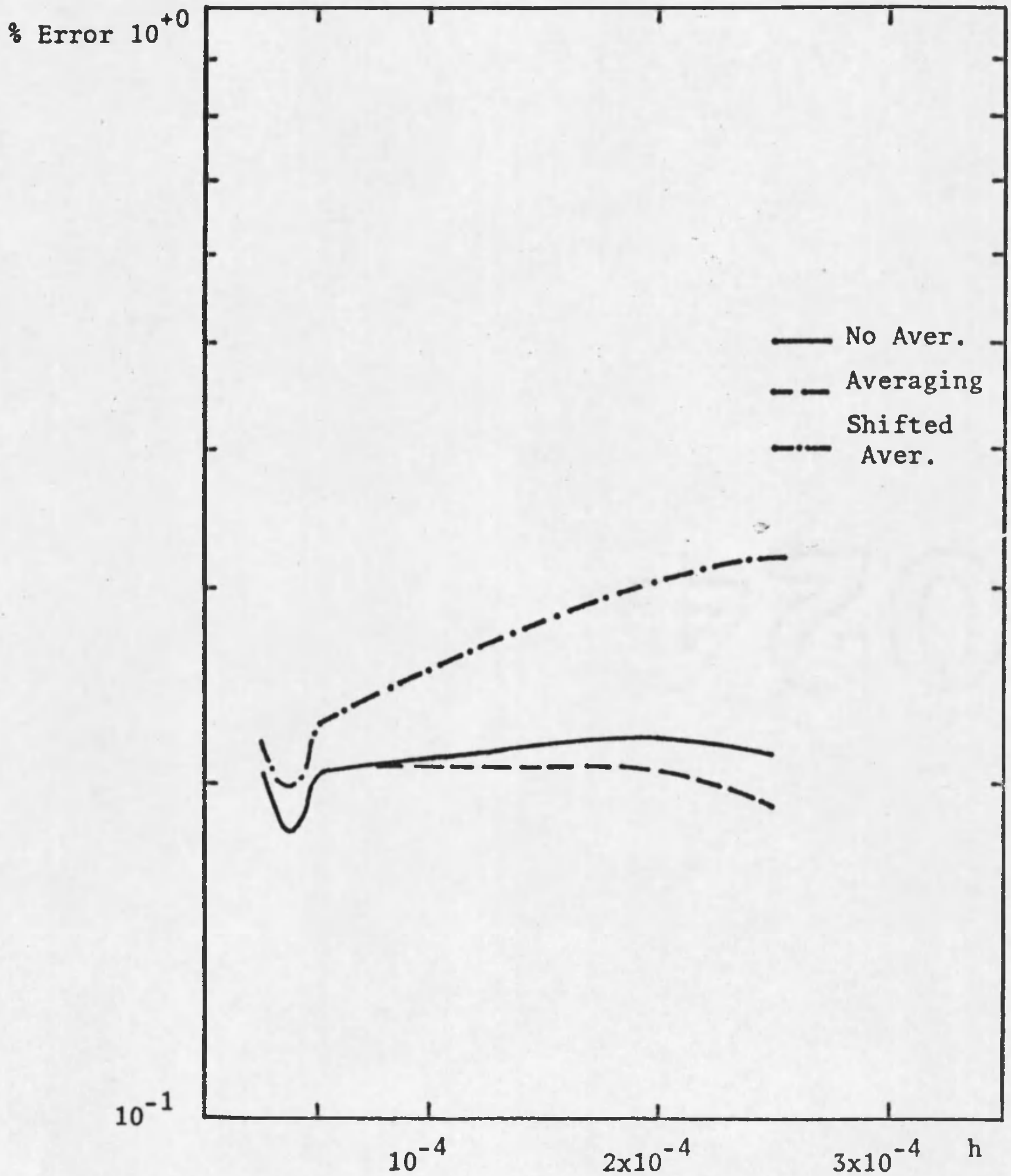

Figure 25. Peak error in X_2 vs. h. Autopilot (partition 2)

method not using averaging. It is obvious then that this method would be an unlikely choice in a simulation.

The Improved Euler method, on the other hand, showed slight improvement in worst-case errors and costs in the pendulum example. This, however, was accompanied by increased avearage peak errors. No real improvement was achieved in the errors of the mine-shaft experiment, but cost was lowered significantly when the Improved Euler method was used. These results therefore indicate that this averaging method is certainly worth considering as an alternate technique to simulate a linear-nonlinear system.

The high cost of and in most cases larger errors obtained from the shifted averaging method used with general nonlinear systems hardly justifies its use. The nonlinear method without averaging is seen to produce significantly lower errors and execution times for both partitions of the autopilot and partition A of the oscillator. The improved errors seen from partition B of the oscillator were at the expense of much higher cost.

The error performance of the simple averaged method is difficult to define. In some experiments, slight improvements were noted, but in others, drastic reductions in accuracy were observed. The inconsistency of performance indicates that further research should be performed to understand why improvements are observed in some cases and not others.

Further experiments using all four methods may also provide insight into when and what type of averaging. The fact that some error improvements were noted show that averaging may prove to be a valuable tool in the simulation of partitioned systems.

APPENDIX A

PROGRAM LISTINGS

This section contains program listings of all algorithms and the two examples that show the use of the two types of simulation methods.

```
C       SUBROUTINES LINKW AND INICON ARE NEEDED BY THE COMBINED
C       ALGORITHMS FOR INITIALIZATION ETC.
C
        SUBROUTINE LINKW(WAV1,WAV2,WAV3,WAV4,WAV5,
     1      WAV6,WAV7,WAV8,WAV9,WAV10)
C
C       THIS SUBROUTINE LINKS OUTPUT VARIABLES AND AVERAGES
C       FROM THE FAST LINEAR SYSTEM TO THE NONLINEAR SYSTEM.
C
        COMMON/LINK/W(10),WAV(10),LINKNO
        COMMON/LINS/LINORD,LININP,LINOUT,A(10,10),B(10,10)
        COMMON/LINS/C(10,10),D(10,10),F(10,10),U(10),X(10)
C
C       LINKNO IS THE NUMBER OF AVERAGES TO BE LINKED
C       AND HAS TO BE DEFINED IN DEFLIN.
C
        GO TO (1,2,3,4,5,6,7,8,9,10),LINKNO
10      WAV10=WAV(10)
9       WAV9=WAV(9)
8       WAV8=WAV(8)
7       WAV7=WAV(7)
6       WAV6=WAV(6)
5       WAV5=WAV(5)
4       WAV4=WAV(4)
3       WAV3=WAV(3)
2       WAV2=WAV(2)
1       WAV1=WAV(1)
        RETURN
        END
        SUBROUTINE INICON
C
C       INICON INITIALIZES MATRICES A,B,C,D, AND F
C       AND VECTORS X,U, AND W
C
        LOGICAL EXIT,RLDONE
        COMMON/LINK/W(10),WAV(10),LINKNO
        COMMON/SYSVAR/DT,DTMAX,DTMIN,EMAX,EMIN,SY(35)
        COMMON/LINS/LINORD,LININP,LINOUT,A(10,10),B(10,10)
        COMMON/LINS/C(10,10),D(10,10),F(10,10),U(10),X(10)
        DIMENSION BN0(10),BN1(10)
C
C       CLEAR ALL MATRICES AND VECTORS
C
        CALL MXCLR1(A,10,10)
        CALL MXCLR1(B,10,10)
        CALL MXCLR1(C,10,10)
        CALL MXCLR1(D,10,10)
        CALL MXCLR1(F,10,10)
        CALL VECCLR(X,10)
        CALL VECCLR(U,10)
        CALL VECCLR(W,10)
        CALL VECCLR(WAV,10)
C
C       INITIALIZE MATRICES AND THE VECTOR X
C
        CALL DEFLIN
C
C       INITIALIZE THE U VECTOR
C
        CALL GFUNC
```

```
C
C       INITIALIZE THE W AND WAV VECTORS
C
        CALL MXCOL(C,X,BN0,LINOUT,LINORD)
        CALL MXCOL(D,U,BN1,LINOUT,LININP)
        CALL VECADD(BN0,BN1,W,LINOUT)
        CALL VECADD(BN0,BN1,WAV,LINOUT)
        RETURN
        END
```

```
      SUBROUTINE INTRGX
C
C      COMBINED ALGORITHM BASED ON IMPROVED EULER METHOD
C
     LOGICAL EXIT,RLDONE,MPRT
      COMMON/LINK/W(10),WAV(10),LINKNO
      COMMON/TVAR/T11,MPRT
     COMMON/SYSVAR/EXIT,RLDONE,IOUT,IFILE,IRUNNO,T,TMAX,TNEXT
     COMMON/SYSVAR/DT,DTMAX,DTMIN,EMAX,EMIN,SY(35)
     COMMON/STATE1/NORDR1,Y(200),GN(200)
     COMMON/LINS/LINORD,LININP,LINOUT,A(10,10),B(10,10)
     COMMON/LINS/C(10,10),D(10,10),F(10,10),U(10),X(10)
     REAL M0(10,10),M1(10,10),M2(10,10)
     DIMENSION GNSAV(200),BN0(10),BN1(10),XAV(10),USAV(10)
      DIMENSION WSAV(10),YSAV(10)
     DIMENSION TRANPA(10,10),BETP0(10,10),BETP1(10,10),BETP2(10,10)
     DIMENSION USUM(10),AVN0(10,10),AVN1(10,10),AVN2(10,10)
      DIMENSION V0(10,10),V1(10,10),D2(10,10),BN2(10),FH(10,10)
     IF(T.GT.0.0) GO TO 30
     IF(DT.GT.DTMAX) DT=DTMAX
     DTO2=DT/2.0
     WRITE(IOUT,20)
20   FORMAT(//1X,35HIMPR EULER WITH CORRECTED AVERAGING)
C
C        1.    P R E C O M P U T A T I O N
C
C
C        COMPUTATION OF D/2 AND H*F
C
      CALL SCALR1(D,0.5,D2,LINOUT,LININP)
      CALL SCALR1(F,DT,FH,LINORD,LINOUT)
C
C       COMPUTATION OF TRANPA,M0,M1,M2
C
       CALL MXCAL(A,TRANPA,M0,M1,M2,LINORD,DT,14,MPRT)
C
C       COMPUTATION OF V0 AND V1
C
       CALL MXEQL(M1,V0,LINORD,LINORD)
       CALL SCALR1(M2,0.5,V1,LINORD,LINORD)
C
C       COMPUTATION OF COEFF. MATRICES FOR AVERAGING
C       VIZ. (AV0+I),(V0-V1)B,V1B
C
       CALL MATMUL(A,V0,AVN1,LINORD,LINORD,LINORD)
       CALL ADDID(AVN1,AVN0,LINORD)
       CALL MXSUB(V0,V1,AVN2,LINORD,LINORD)
       CALL MATMUL(AVN2,B,AVN1,LINORD,LINORD,LININP)
       CALL MATMUL(V1,B,AVN2,LINORD,LINORD,LININP)
C
C       COMPUTATION OF COEFF. MATRICES FOR LINEAR DISCRETE VALUES
C       VIZ. (AM0+I),(M0-M1)B,M1B
C
       CALL MATMUL(A,M0,BETP1,LINORD,LINORD,LINORD)
       CALL ADDID(BETP1,BETP0,LINORD)
       CALL MXSUB(M0,M1,BETP2,LINORD,LINORD)
       CALL MATMUL(BETP2,B,BETP1,LINORD,LINORD,LININP)
       CALL MATMUL(M1,B,BETP2,LINORD,LINORD,LININP)
       CALL INICON
C
C
```

```
C       2.  R U N   T I M E   C O M P U T A T I O N
C
C
C       4. COMPUTATICN OF Y(N+1) AND U(N+1)
30      RLDCNE = .FALSE.
        DO 35 J=1,NORDR1
        YSAV(J)=Y(J)
        Y(J)=Y(J)+DT*GN(J)
        GNSAV(J)=GN(J)
35      CONTINUE
        DO 40 J=1,LININP
        USAV(J)=U(J)
40      CONTINUE
        T=T+DT
        CALL DIFEQ1
        DO 50 J=1,NORDR1
        Y(J)=YSAV(J)+DTO2*(GNSAV(J)+GN(J))
50      CONTINUE
        CALL GFUNC
C
C       COMPUTATICN OF XAV(N+1)
C
        DO 60 J=1,LINOUT
        WSAV(J)=WAV(J)
60      CONTINUE
        CALL MXCOL(AVNO,X,BNO,LINCRD,LINCRD)
        CALL MXCOL(AVN1,USAV,BN1,LINORD,LININP)
        CALL VECADD(BNO,BN1,BN2,LINCRD)
        CALL MXCOL(AVN2,U,BN1,LINORD,LININP)
        CALL VECADD(BN1,BN2,XAV,LINCRD)
C
C       COMPUTATICN CF WAV(N+1)
C
        CALL MXCOL(C,XAV,BNO,LINCUT,LINCRD)
        CALL VECADD(USAV,U,USUM,LININP)
        CALL MXCCL(D2,USUM,BN1,LINCUT,LININP)
        CALL VECADD(BNO,BN1,WAV,LINOUT)
C
C       ADDITICN OF CORRECTICN FACTOR TO Y(N+1)
C
        DO 70 J=1,NCRDR1
        BN2(J)=Y(J)
70      CONTINUE
        CALL VECSUB(WAV,WSAV,BNO,LINOUT)
        CALL MXCOL(FH,BNO,BN1,LINORD,LINCUT)
        CALL VECADD(BN2,BN1,Y,LINORD)
        CALL GFUNC
C
C       COMPUTATICN CF X(N+1)
C
        CALL MXCOL(BETPO,X,BNO,LINORD,LINCRD)
        CALL MXCOL(BETP1,USAV,BN1,LINCRD,LININP)
        CALL VECADD(BNO,BN1,BN2,LINCRD)
        CALL MXCOL(BETP2,U,BN1,LINORD,LININP)
        CALL VECADD(BN2,BN1,X,LINORD)
C
C       COMPUTATION CF W
C
        CALL MXCOL(C,X,BNO,LINCUT,LINCRD)
        CALL MXCCL(D,U,BN1,LINOUT,LININP)
```

```
      CALL VECADD(BNO,BN1,W,LINOUT)
      RLDONE=.TRUE.
      CALL DIFEQ1
      RETURN
      END
```

```
      SUBROUTINE INTRGX
C
C     COMBINED ALGORITHM BASED ON MODIFIED EULER
C
      LOGICAL EXIT,RLDONE,MPRT
      COMMON/LINK/W(10),WAV(10),LINKNO
      COMMON/TVAR/I11,MPRT
      COMMON/SYSVAR/EXIT,RLDONE,IOUT,IFILE,IRUNNO,T,TMAX,TNEXT
      COMMON/SYSVAR/DT,DTMAX,DTMIN,EMAX,EMIN,SY(35)
      COMMON/STATE1/NORDR1,Y(200),GN(200)
      COMMON/LINS/LINORD,LININP,LINOUT,A(10,10),B(10,10)
      COMMON/LINS/C(10,10),D(10,10),F(10,10),U(10),X(10)
      REAL M0(10,10),M1(10,10),M2(10,10),M3(10,10)
      DIMENSION GNSAV(200),BN0(10),BN1(10),XAV(10),USAV(10)
      DIMENSION V2(10,10),AVN3(10,10),AVN4(10,10),USAV2(10)
      DIMENSION BETP3(10,10),BETP4(10,10)
      DIMENSION WSAV(10),YSAV(10),BN3(10)
      DIMENSION TRANPA(10,10),BETP0(10,10),BETP1(10,10),BETP2(10,10)
      DIMENSION USUM(10),AVN0(10,10),AVN1(10,10),AVN2(10,10)
      DIMENSION V0(10,10),V1(10,10),D6(10,10),BN2(10),FH(10,10)
      IF(T.GT.0.0) GO TO 30
      IF(DT.GT.DTMAX) DT=DTMAX
      DTO2=DT/2.0
      WRITE(IOUT,20)
 20   FORMAT(//1X,35HMOD EULER WITH CORRECTED AVERAGING)
C
C        P R E C O M P U T A T I O N
C
C
C     COMPUTATION OF D/6 AND H*F
C
      CALL SCALR1(D,1./6.,D6,LINOUT,LININP)
      CALL SCALR1(F,DT,FH,LINORD,LINOUT)
C
C     COMPUTATION OF TRANPA,M0,M1,M2,M3
C
      CALL MXCAL3(A,TRANPA,M0,M1,M2,M3,LINORD,DT,14,MPRT)
C
C     COMPUTATION OF V0 ,V1 AND V2
C
      CALL MXEQL(M1,V0,LINORD,LINORD)
      CALL SCALR1(M2,0.5,V1,LINORD,LINORD)
      CALL SCALR1(M3,1./3.,V2,LINORD,LINORD)
C
C     COMPUTATION OF COEFF. MATRICES FOR AVERAGING
C     VIZ. (AV0+I),(V0-3V1+2V2)B,
C     4(V1-V2)B,(2V2-V1)B
C
      CALL MATMUL(A,V0,AVN1,LINORD,LINORD,LINORD)
      CALL ADDID(AVN1,AVN0,LINORD)
      CALL SCALR1(V1,-3.0,AVN1,LINORD,LINORD)
      CALL SCALR1(V2,2.0,AVN4,LINORD,LINORD)
      CALL MATADD(V0,AVN1,AVN1,LINORD,LINORD)
      CALL MATADD(AVN1,AVN4,AVN4,LINORD,LINORD)
      CALL MATMUL(AVN4,B,AVN1,LINORD,LINORD,LININP)
      CALL MXSUB(V1,V2,AVN4,LINORD,LINORD)
      CALL SCALR1(AVN4,4.0,AVN4,LINORD,LINORD)
      CALL MATMUL(AVN4,B,AVN2,LINORD,LINORD,LININP)
      CALL SCALR1(V2,2.0,AVN4,LINORD,LINORD)
      CALL MXSUB(AVN4,V1,AVN4,LINORD,LINORD)
```

```
60      CONTINUE
        CALL MXCOL(C,XAV,BNO,LINOUT,LINORD)
        CALL SCALRT(USAV2,4.0,BN1,LININP,LININP)
        CALL VECADD(USAV,BN1,BN2,LININP)
        CALL VECADD(BN2,U,USUM,LININP)
        CALL MXCOL(D6,USUM,BN1,LINOUT,LININP)
        CALL VECADD(BNO,BN1,WAV,LINOUT)
C
C       ADDITION OF CORRECTION FACTOR TO Y(N+1)
C
        DO 70 J=1,NORDR1
        BN2(J)=Y(J)
70      CONTINUE
        CALL VECSUB(WAV,WSAV,BNO,LINOUT)
        CALL MXCOL(FH,BNO,BN1,LINORD,LINOUT)
        CALL VECADD(BN2,BN1,Y,LINORD)
        CALL GFUNC
C
C       COMPUTATION OF X(N+1)
C
        CALL MXCOL(BETP0,X,BNO,LINORD,LINORD)
        CALL MXCOL(BETP1,USAV,BN1,LINORD,LININP)
        CALL VECADD(BNO,BN1,BN2,LINORD)
        CALL MXCOL(BETP2,USAV2,BNO,LINORD,LINORD)
        CALL VECADD(BNO,BN2,BN2,LINORD)
        CALL MXCOL(BETP3,U,BNO,LINORD,LININP)
        CALL VECADD(BNO,BN2,X,LINORD)
C
C       COMPUTATION OF W
C
        CALL MXCOL(C,X,BNO,LINOUT,LINORD)
        CALL MXCOL(D,U,BN1,LINOUT,LININP)
        CALL VECADD(BNO,BN1,W,LINOUT)
        RLDONE=.TRUE.
        CALL DIFEQ1
        RETURN
        END
```

```
      SUBROUTINE INTRGX
C
C
C      NONLINEAR ALGORITHM WITH AVERAGING
C
C
      LOGICAL EXIT,RLDONE
      COMMCN/SYSVAR/EXIT,RLDCNE,ICUT,IFILE,IRUNNO,T,TMAX,DUMM
      COMMON/SYSVAR/DT,DTMAX,DTMIN,EMAX,EMIN,SY(35)
      COMMCN/STATE1/NORDR1,Y(200),GN(200)
      COMMON/STATE2/NORDR2,Y2(200),GN2(200)
     COMMCN/AVER/YAV(200)
      DIMENSICN GTEM(200),YTEM(200)
      REAL K1(200),K2(200),K3(300)
      IF(T.GT.0.0) GO TO 30
C
C     ****** REMINDER :  DTMAX.LE.TMAX/(NPOINT-1)
C
      IF(DT.GT.DTMAX) DT=DTMAX
      DTA=DT
      DT2=DT/2.0
C
C    DTMAX = DT2   SET IF  DTMAX.GT.DT2    IN ORDER TO AVOID
C    HANG UP ON DTMAX IN RKM  -   SEE COMMENTS IM RKM SUBR.
C
       IF(DTMAX.GT.DT2) DTMAX = DT2
     WRITE(IOUT,11) DTMAX
 11  FORMAT(14H NEW   DTMAX =,E12.5/)
      DT3=DT/3.0
      DT4=DT/4.0
      DT6=DT/6.0
      WRITE(IOUT,1)
  1   FORMAT(/52H PARTITIONED INTEGRATION-   R K - 4  FOR SLOW SYSTEM)
 30     RLDCNE=.FALSE.
C     FAST SYSTEM INTEGRATICN ( NH<= T <=(N+1/2)H ) USING RKM
      TEMP=T
      TNEXT=T+DT2
C
C  DT4  SET INSTEAD DT  AT  T=0.0  TO START STEP CONTROL
C  IN RKM SUBROUTINE
C
      IF(T.EQ.0.0) DT=DT4
C
C*****  ZERO AVERAGES FOR HALF STEP INTERVAL
C
     DO 35 II=1,NORDR1
35   YAV(II)=0.0
40   T1=T
      CALL RKM(TEMP,TNEXT)
     DLT=T-T1
C    COMPUTE AVERAGES
     DO 45 II=1,NORDR1
45   YAV(II)=YAV(II)+DLT*Y(II)
      IF(TNEXT.GT.T) GO TO 40
     DO 46 II=1,NOBDR1
46   YAV(II)=YAV(II)/DT2
C IN GENERAL NEXT STEP WOULD BE COMPUTATICN CF U(N+1/2)
C  2.    SLOW SYSTEM INTEGRATION  -  SECOND ORDER IMPROVED EULER
C  2.1. COMPUTATICN OF K1,K2,K3
      DO 100 I=1,NORDR2
      K1(I)=GN2(I)
```

```
       YTEM(I)=Y2(I)
  100  Y2(I)=Y2(I)+DT2*K1(I)
       CALL DIFEQ2
       DO 110 I=1,NORDR2
       K2(I)=GN2(I)
 110   Y2(I)=YTEM(I)+DT2*K2(I)
       CALL DIFEQ2
       DO 120 I=1,NORDR2
 120   K3(I)=GN2(I)
C
C   2.2. COMPUTATION OF Y(N+1/2) AND EVALUATICN OF SLOW SYSTEM EQU.

       DO 130 I=1,NCRDR2
 130   Y2(I)=YTEM(I)+DT4*(K1(I)+K2(I))
       CALL DIFEQ2
C
C    3.  FAST SYSTEM IINTEGRATICN
C
       TEMP=T
       TNEXT=T+DT2
C
C*****  ZERO AVERAGES FOR FULL STEP INTERVAL
C
       DO 47 II=1,NORDR1
47     YAV(II)=0.0
50     T1=T
        CALL RKM(TEMP,TNEXT)
       DLT=T-T1
C
C*****  COMPUTE AVERAGES FOR FULL STEP
       DO 55 II=1,NCRDR1
55     YAV(II)=YAV(II)+DLT*Y(II)
        IF(TNEXT.GT.T) GO TO 50
       DO 56 II=1,NORDR1
56     YAV(II)=YAV(II)/DT2
       CALL DIFEQ2
C       IN GENERAL NEXT STEP WOULD EE COMPUTATION OF U(N+1)
C
C    4.  COMPUTATION OF K4 AND X(N+1)
        DO 200 I=1,NCRDR2
  200   Y2(I)=YTEM(I)+DTA*K3(I)
        CALL DIFEQ2
        DO 210 I=1,NORDR2
  210   Y2(I)=YTEM(I)+DT6*(K1(I)+GN2(I))+DT3*(K2(I)+K3(I))
C
C    NEXT STEP PREPARATICN
        RLDONE=.TRUE.
        CALL DIFEQ2
        RETURN
        END
```

```
      SUBROUTINE INTRGX
C
C     NONLINEAR ALGORITHM WITH SHIFTED AVERAGING
C
      LOGICAL EXIT,RLDONE
      COMMON/SYSVAR/EXIT,RLDONE,IOUT,IFILE,IRUNNO,T,TMAX,DUMM
      COMMON/SYSVAR/DT,DTMAX,DTMIN,EMAX,EMIN,SY(35)
      COMMON/STATE1/NORDR1,Y(200),GN(200)
      COMMON/STATE2/NORDR2,Y2(200),GN2(200)
     COMMON/AVER/YAV(200)
      DIMENSION GTEM(200),YTEM(200),YSAV(200),YSAV2(200)
     DIMENSION AV(200),YAVSV(200)
      REAL K1(200),K2(200),K3(300)
      IF(T.GT.0.0) GO TO 30
C
C      ****** REMINDER :  DTMAX.LE.TMAX/(NPOINT-1)
C
      IF(DT.GT.DTMAX) DT=DTMAX
      DTA=DT
      DT2=DT/2.0
C
C     DTMAX = DT2   SET IF  DTMAX.GT.DT2    IN ORDER TO AVOID
C     HANG UP ON DTMAX IN RKM  -    SEE COMMENTS IN RKM SUBR.
C
       IF(DTMAX.GT.DT2) DTMAX = DT2
     WRITE(IOUT,11) DTMAX
 11  FORMAT(14H NEW   DTMAX =,E12.5/)
      DT3=DT/3.0
      DT4=DT/4.0
      DT6=DT/6.0
      WRITE(IOUT,1)
  1   FORMAT(/52H PARTITIONED INTEGRATION-   R K - 4  FOR SLOW SYSTEM)
 30   RLDONE=.FALSE.
C     FAST SYSTEM INTEGRATION ( NH<= T <=(N+1/2)H ) USING RKM
      TEMP=T
      TNEXT=T+DT4
C
C   DT4  SET INSTEAD DT  AT  T=0.0  TO START STEP CONTROL
C   IN RKM SUBROUTINE
C
      IF(T.EQ.0.0) DT=DT4/2.
C
C***   FIRST QUARTER STEP
C
40    CALL RKM(TEMP,TNEXT)
      IF(TNEXT.GT.T)GO TO 40
      TEMP=T
      TNEXT=T+DT4
C
C***   SECOND QUARTER STEP - START AVERAGING
C
      DO 41 II=1,NORDR1
41    AV(II)=0.0
42    T1=T
       CALL RKM(TEMP,TNEXT)
      DLT=T-T1
C
C***   COMPUTE AVERAGE
C
      DO 43 II=1,NORDR1
43    AV(II)=AV(II)+DLT*Y(II)
```

```
         IF(TNEXT.GT.T) GO TO 42
        DO 44 II=1,NORDR1
        YSAV(II)=Y(II)
44      YAV(II)=Y(II)
C IN GENERAL NEXT STEP WOULD BE COMPUTATION OF U(N+1/2)
C  2.     SLOW SYSTEM INTEGRATION  -  SECOND ORDER IMPROVED EULER
C  2.1. COMPUTATION OF K1,K2,K3
         DO 100 I=1,NORDR2
         K1(I)=GN2(I)
         YTEM(I)=Y2(I)
  100    Y2(I)=Y2(I)+DT2*K1(I)
         CALL DIFEQ2
         DO 110 I=1,NORDR2
         K2(I)=GN2(I)
C
C   2.2. COMPUTATION OF APPROX. Y(N+1/2)

         DO 130 I=1,NORDR2
 130     Y2(I)=YTEM(I)+DT4*(K1(I)+K2(I))
        CALL DIFEQ2
        TEMP=T
        TNEXT=T+DT4
C
C***    THIRD QUARTER STEP - COMPLETE AVERAGE
C
46      T1=T
        CALL RKM(TEMP,TNEXT)
        DLT=T-T1
        DO 49 II=1,NORDR1
        AV(II)=AV(II)+DLT*Y(II)
49      YAV(II)=YAVSV(II)
        IF(TNEXT.GT.T)GO TO 46
        T=TEMP
        DO 47 II=1,NCRDR1
47      Y(II)=YSAV(II)
        DO 48 II=1,NORDR2
48      Y2(II)=YTEM(II)
        CALL DIFEQ2
        DO 501 II=1,NORDR1
501     YAV(II)=AV(II)/DT2
         DO 150 I=1,NORDR2
         K1(I)=GN2(I)
         YTEM(I)=Y2(I)
C
C***    RECOMPUTE K1 AND K2, FIND K3
C
  150    Y2(I)=Y2(I)+DT2*K1(I)
         CALL DIFEQ2
         DO 160 I=1,NORDR2
         K2(I)=GN2(I)
 160     Y2(I)=YTEM(I)+DT2*K2(I)
         CALL DIFEQ2
         DO 170 I=1,NORDR2
 170     K3(I)=GN2(I)
C
C   2.2. COMPUTATION OF Y(N+1/2) AND EVALUATION OF SLOW SYSTEM EQU.

         DO 180 I=1,NORDR2
 180     Y2(I)=YTEM(I)+DT4*(K1(I)+K2(I))
        CALL DIFEQ2
C
```

```
C     3.  FAST SYSTEM IINTEGRATICN
C
C
C*** DO THIRD QUARTER AGAIN
C
       TEMP=T
       TNEXI=T+DT4
50     CALL RKM(TEMP,TNEXT)
       IF(TNEXT.GT.T)GO TO 50
C
C***   FOURTH QUARTER - START AVERAGING
C
        TEMP=T
        TNEXT=T+DT4
       DO 51 II=1,NORDR1
51     AV(II)=0.0
52     T1=T
        CALL RKM(TEMP,TNEXT)
       DLT=T-T1
       DO 53 II=1,NORDR1
53     AV(II)=AV(II)+DLT*Y(II)
        IF(TNEXT.GT.T) GO TO 52
       DO 54 II=1,NORDR2
54     YSAV2(II)=Y2(II)
       DO 55 II=1,NORDR1
       YSAV(II)=Y(II)
55     YAV(II)=Y(II)
       CALL DIFEQ2
C       IN GENERAL NEXT STEP WOULD BE COMPUTATION OF U(N+1)
C
C     4.  COMPUTATION OF K4 AND APPROX. Y(N+1)
         DO 200 I=1,NORDR2
  200    Y2(I)=YTEM(I)+DTA*K3(I)
         CALL DIFEQ2
         DO 210 I=1,NORDR2
  210    Y2(I)=YTEM(I)+DT6*(K1(I)+GN2(I))+DT3*(K2(I)+K3(I))
       CALL DIFEQ2
C
C***   FIFTH QUARTER - COMPLETE AVERAGING
C
       TEMP=T
       TNEXT=T+DT4
56     T1=T
       CALL RKM(TEMP,TNEXT)
       DLT=T-T1
       DO 59 II=1,NORDR1
       AV(II)=AV(II)+DLT*Y(II)
59     YAV(II)=AV(II)/DT/2
       IF(TNEXT.GT.T)GO TO 56
       T=TEMP
       DO 57 II=1,NCRDR1
57     Y(II)=YSAV(II)
       DO 58 II=1,NORDR2
58     Y2(II)=YSAV2(II)
       CALL DIFEQ2
C
C***   RECOMPUTE K4 AND FIND FINAL Y(N+1)
         DO 250 I=1,NCRDR2
  250    Y2(I)=YTEM(I)+DTA*K3(I)
         CALL DIFEQ2
         DO 260 I=1,NORDR2
```

```
  260       Y2(I)=YTEM(I)+DT6*(K1(I)+GN2(I))+DT3*(K2(I)+K3(I))
C
C     NEXT STEP PREPARATION
        DO 270 II=1,NORDR1
270     YAVSV(II)=YAV(II)
          RLDONE=.TRUE.
        CALL DIFEQ1
          CALL DIFEQ2
          RETURN
          END
```

```
      SUBROUTINE RKM(TEMP,TNEXT)
C
C PROGRAM.. INTRGX SUBROUTINE
C
C RUNGE-KUTTA-MERSON RULE MODIFIED FOR PARTITIONED INTEGRATION
C     (COMMMENTS MARKED WITH **** )
C             UNIVERSITY OF ARIZONA   , MAY 1977
C             OLGIERD A. PALUSINSKI
C             TECHNICAL UNIVERSITY OF SILESIA
C             44-100 GLIWICE,POLAND
C
C
C TYPE OF PROGRAM.. RUNGE-KUTTA-MERSON VARIABLE STEP INTEGRATION
C                   RULE FOR INTEGRATING ORDINARY DIFFERENTIAL
C                   EQUATIONS.
C
C VERSION AND DATE..   4.0    MAY 1976
C
C AUTHOR.. JOHN V. WAIT
C          ELECTRICAL ENGINEERING DEPARTMENT
C          UNIVERSITY OF ARIZONA
C          TUCSON, ARIZONA 85721
C
C   MODIFICATIONS OF ERROR CONTROL LOGIC INCLUDED
C       ( COMMENTS MARKED WITH %%%%%% )
C
C
C LANGUAGE.. ANSI STANDARD FORTRAN IV
C
C ABSTRACT..
C
C   SEE -APPLIED NUMERICAL METHODS- BY CARNAHAN, LUTHER, AND WILKES
C   JOHN WILEY AND SONS, INC, 1969, NEW YORK, LONDON, SYDNEY, TORONTO
C     (A GENERAL DISCUSSION IS GIVEN OF RUNGE-KUTTA RULES AND SPECIFIC
C     DISCUSSIONS ARE GIVEN FOR THE RUNGE-KUTTA SECOND, THIRD, AND
C     FOURTH ORDER SYSTEMS.  HOWEVER, THE RUNGE-KUTTA-MERSON METHOD
C     IS NOT DISCUSSED IN PARTICULAR.)
C
C   IF SY(1) .GE. 1.0, USE ABSOLUTE TEST ON..    Y(IFIX(SY(1)))
C   IF SY(1) .LT. 1.0, USE RELATIVE ERROR ON ..    ALL Y
C   IN SY(2) IS KEPT THE MAXIMUM DEADLOCKED ERROR
C   IF SY(6) .GT. 0.0, OUTPUT CHANGES TO DT
C
C
C-----------------------------------------------------------------------
C
C$ THIS IS THE SINGLE PRECISION VERSION OF RULE01.
C
C SUBROUTINE CONVRT WILL CONVERT THE PRECISION OF THIS SUBROUTINE
C SEE SUBROUTINE CONVRT OR SUBROUTINE RULE11 FOR AN EXPLANATION
C OF THE ORDERING AND FLAGGING CONVENTIONS USED IN THIS
C CONVERTABLE SUBROUTINE
C
C-----------------------------------------------------------------------
C
      LOGICAL A,B,FO,X,LIST
C(2
CD    DOUBLE PRECISION RK1(200),RK3(200),RK4(200),RK5(200)
CD    DOUBLE PRECISION YOLD(200),DPY(200),DPT,TIME
C)1
      REAL RK1(200),RK3(200),RK4(200),RK5(200),YOLD(200)
```

```
C/5
      LOGICAL EXIT,RLDONE
      COMMON /SYSVAR/ EXIT,RLDONE,IOUT,IFILE,IRUNNO,T,TMAX,DUMM
      COMMON /SYSVAR/ DT,DTMAX,DTMIN,EMAX,EMIN,SY(35)
      COMMON /STATE1/ NORDR1,Y(200),GN(200)
C *************** WATCH OUT ************************
       COMMON/STATE2/ NORDR2,Y2(200),GN2(200)
       COMMCN/EXTR/YEXTR(200),W(10)
C ****************************************************
C
C  INITIALIZE
      A=.TRUE.
      IF(T.GT.0.) GO TO 5
C(3
CD    DO 1 I=1,NORDR1
CD1     DPY(I)=Y(I)
CD    DPT=0.D0
C)0
C/5
      IF(IFIX(SY(1)).LE.NORDR1) GO TO 3
      WRITE(IOUT,2)
2     FORMAT(48H WARNING - SY(1) TOO LARGE FOR EQUATIONS GIVEN.
     2 17HSY(1) SET TO ZERO)
      SY(1)=0.
3     SY(2)=EMAX
      LIST=SY(6).NE.0.
      WRITE(IOUT,4)
4     FORMAT(52H RUNGE-KUTTA-MERSCN INTEGRATICN RULE FOR FAST SYSTEM)
C
C  MESSAGE FOR PARTITIONED INTEGRATION ONLY **************
C
      WRITE(IOUT,444) IRUNNO
 444  FORMAT(33H LINEAR EXTRAPOLATION , RUN NO.  ,I3/)
C
C  ******************************************************
C
      ISY1=SY(1)
5     FO=T.LT.TNEXT.AND.TNEXT.LT.T+DT
      DTEM=DT
      IF(FO) DT=TNEXT-T
C(2
CD    DT3=DBLE(DT)/3.D0
CD    TIME=DPT
C)2
      DT3=DT/3.
      TIME=T
C/6
C  FIND K1
C  DIFEQ1 WAS CALLED BEFORE ENTRY
      RLDONE=.FALSE.
      DO 7 I=1,NORDR1
C(4
CD       RK1(I)=DBLE(GN(I))*DT3
CD       YOLD(I)=DPY(I)
CD       DPY(I)=DPY(I)+RK1(I)
CD       Y(I)=SNGL(DPY(I))
C)3
         RK1(I)=GN(I)*DT3
         YOLD(I)=Y(I)
         Y(I)=Y(I)+RK1(I)
C/9
```

```
7         CONTINUE
C(2
CD      DPT=TIME+DT3
CD      T=SNGL(DPT)
C)1
        T=TIME+DT3
C/5
C  FIND K2
C ****************** WATCH OUT ***************** LINEAR EXTRAPOLATION
          DELTA=T-TEMP
          DO 991 I=1,NORDR2
 991      YEXTR(I)=Y2(I)+GN2(I)*DELTA
C*****************************************
        CALL DIFEQ1
        DO 8 I=1,NORDR1
C(2
CD        DPY(I)=YOLD(I)+.5D0*(RK1(I)+DBLE(GN(I))*DT3)
CD        Y(I)=SNGL(DPY(I))
C)1
          Y(I)=YOLD(I)+0.5*(RK1(I)+GN(I)*DT3)
C/5
8         CONTINUE
C  FIND K3
        CALL DIFEQ1
        DO 9 I=1,NORDR1
C(3
CD        RK3(I)=4.5D0*DT3*DBLE(GN(I))
CD        DPY(I)=YOLD(I)+.375D0*RK1(I)+.25D0*RK3(I)
CD        Y(I)=SNGL(DPY(I))
C)2
          RK3(I)=4.5*DT3*GN(I)
          Y(I)=YOLD(I)+0.375*RK1(I)+0.25*RK3(I)
C/7
9         CONTINUE
C(2
CD      DPT=TIME+.5D0*DBLE(DT)
CD      T=SNGL(DPT)
C)1
        T=TIME+.5*DT
C/5
C  FIND K4
C ************ WATCH OUT **************** LINEAR EXTRAPOLATION ***
          DELTA=T-TEMP
          DO 992 I=1,NORDR2
 992      YEXTR(I)=Y2(I)+GN2(I)*DELTA
C ***************************************
        CALL DIFEQ1
        DO 10 I=1,NORDR1
C(3
CD        RK4(I)=4.D0*DT3*DBLE(GN(I))
CD        DPY(I)=YOLD(I)+1.5D0*(RK1(I)+RK4(I))-RK3(I)
CD        Y(I)=SNGL(DPY(I))
C)2
          RK4(I)=4.*DT3*GN(I)
          Y(I)=YOLD(I)+1.5*(RK1(I)+RK4(I))-RK3(I)
C/7
10        CONTINUE
C(2
CD      DPT=TIME+DBLE(DT)
CD      T=SNGL(DPT)
```

```
C) 1
      T=TIME+DT
C/5
C   FIND K5 AND NEXT POINT
C *************WATCH OUT ***************** LINEAR EXTRAPOLATICN **
         DELTA=T-TEMP
         DO 993 I=1,NORDR2
 993     YEXTR(I)=Y2(I)+GN2(I)*DELTA
C *****************************************
      CALL DIFEQ1
      DO 11 I=1,NORDR1
C(3
CD       RK5(I)=DBLE(GN(I))*DT3
CD       DPY(I)=YOLD(I)+.5D0*(RK1(I)+RK4(I)+RK5(I))
CD       Y(I)=SNGL(DPY(I))
C) 2
         RK5(I)=GN(I)*DT3
         Y(I)=YOLD(I)+.5*(RK1(I)+RK4(I)+RK5(I))
C/7
11       CONTINUE
C
C  %%%%  MODIFICATION OF LOGIC : BYPASS  "C" PUT IN COL.1 IN NEXT ST.
C                THIS MEANS ERROR IS CHECKED ALWAYS(REGARDLESS OF "FO")
C
C       IF(FO) GO TO 20
C
C  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
C
C FIND ERROR
C
      IF(ISY1.GT.0) GO TO 13
C
C SY(1)=0, DC RELATIVE ERROR CHECK
C
      ERROR=0.
      DO 12 I=1,NORDR1
C(2
CD       RK6=(RK1(I)-RK3(I)+RK4(I))*.2D0-RK5(I)*.1D0
CD    WEIGHT=ABS(Y(I))+ABS(SNGL(YOLD(I))-Y(I))+1.
C) 2
      RK6=(RK1(I)-RK3(I)+RK4(I))*.2-RK5(I)*.1
      WEIGHT=ABS(Y(I))+ABS(YOLD(I)-Y(I))+1.
C/6
      ERRCR=AMAX1(ERFOR,ABS(RK6)/WEIGHT)
12    CONTINUE
      GO TO 14
C
C SY(1).GT.0, DO ABSCLUTE TEST CN Y(IFIX(SY(1))
C
C(2
CD13  ERROR=SNGL(DABS((RK1(ISY1)-RK3(ISY1)+RK4(ISY1))*.2D0
CD   2          -RK5(ISY1)/.1D0))
C) 1
  13  ERROR=ABS((RK1(ISY1)-RK3(ISY1)+RK4(ISY1))*.2-RK5(ISY1)*.1)
C/5
C
C TEST ERROR
C
14    B=(ERROR.GE.EMIN).CR.(DT.GE.DTMAX).CR.(.NCT.A)
      A=(ERROR.LE.EMAX).CR.(DT.LE.DTMIN)
```

```
      X=(ERROR.LE.SY(2)).OR.(DT.GT.DTMIN)
C
C IF X=.FALSE., DEADLOCK AND ERROR IS GREATER THAN BEFORE
C
      IF(X) GO TO 151
        WRITE(IOUT,15) ERROR,T,DT
15      FORMAT(20H  DEADLOCK - ERROR =,1PE14.7,5H  T =,
     2   E14.7,6H  DT =,E14.7)
        SY(2)=ERROR
        GO TO 19
C
C IF A=.FALSE., HALVE DT
C
C %%% MODIFICATION OF LOGIC : #222 PUT INSTEAD OF #18 IN NEXT ST.
C              THIS MEANS THAT BEFORE "B" THE VALUE OF "FO" IS CHECKED
C
151   IF(A) GO TO 222
C
C  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
C(2
CD      DPT=TIME
CD      T=SNGL(DPT)
C)1
        T=TIME
C/5
        DT=AMAX1(DT/2.,DTMIN)
        IF(LIST) WRITE(IOUT,16) DT,T
16      FORMAT(10H  NEW DT =,1PE14.7,5H  T =,E14.7)
        DO 17 I=1,NORDR1
C(3
CD        DPY(I)=YOLD(I)
CD        Y(I)=SNGL(DPY(I))
CD17      CONTINUE
C)1
  17      Y(I)=YOLD(I)
C/6
C ***************WATCH OUT ****************** LINEAR EXTRAPOLATION **
          DELTA=T-TEMP
          DO 994 I=1,NORDR2
  994     YEXTR(I)=Y2(I)+GN2(I)*DELTA
C *********************************************
        CALL DIFEQ1
        GO TO 5
C
C  %%% MODIFICATION OF LOGIC : NEW STATEMENT TO CHECK VALUE OF "FO"
C      IN ORDER TO PREVENT DOUBLING WHEN "FO"=.TRUE.
C
 222   IF(FO) GO TO 20
C
C  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
C
C
C IF B=.FALSE., DOUBLE DT
C
18    IF(B) GO TO 19
C
C ********REMINDER:****** DT2  PUT IN PLACE OF  DTMAX  WHEN
C                     DTMAX.GT.DT2   IN ORDER TO PREVENT
C                      HANG UP ON DTMAX ;
```

```
C                        THAT MEANS    DTMAX = DT2   FOR RKM ERROR CONTROL
C                         IF    DTMAX.GT.DT2 ; SEE "INTRGX"   FOR   T=0.
C
        DT=AMIN1(DT*2.,DTMAX)
C
C*******************************************************************
C
        IF(LIST) WRITE(IOUT,16) DT,T
19     RLDONE=.TRUE.
C *************** WATCH OUT ****************** LINEAR EXTRAPOLATION **
             DELTA=T-TEMP
             DO 995 I=1,NORDR2
  995        YEXTR(I)=Y2(I)+GN2(I)*DELTA
C ********************************************
      CALL DIFEQ1
      RETURN
C
C FO=.TRUE., RESTORE DT
C
20    DT=DTEM
C  %%%%%%% MODIF. OF LOGIC: "C" PUT IN COL.1 IN NEXT TWO ST.
C       A=.TRUE.
C       B=.TRUE.
C  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      RLDONE=.TRUE.
C ************* WATCH OUT *************** LINEAR EXTRAPOLATION ***
          DELTA=T-TEMP
          DO 996 I=1,NORDR2
 996      YEXTR(I)=Y2(I)+GN2(I)*DELTA
C ****************************************
      CALL DIFEQ1
      RETURN
      END
```

```
      SUBROUTINE MXCAL3(A,PHIOFT,M0,M1,M2,M3,ID,DT,NDIG,PT)
C
C        MXCAL3 FINDS M0,M1,M2,M3 OF A MATRIX A
C
C***     ADD DIMENSION OF M3 AND M33
C
      REAL M0(10,10), M1(10,10), M2(10,10), M3(10,10), M00(10,10),
     $M11(10,10), M22(10,10), M33(10,10), MPHI(10,10)
      DIMENSION A(10,10), PHIOFT(10,10)
      INTEGER SIGDIG(14)
      LOGICAL PT
      DATA (SIGDIG(I),I=1,14)/4,6,7,8,9,10,11,12,13,14,
     $15,15,16,17/
      AIJMAX = 10.0**(-NDIG)
      DO 05 I=1,ID
      DO 05 J=1,ID
      TEMP = ABS(A(I,J))
   05 IF(TEMP.GT.AIJMAX) AIJMAX = TEMP
      KOUNT = 1
      DTMAX = 1.0 / AIJMAX
      TFRAME = DT / 2.0
   10 IF(TFRAME.LE.DTMAX) GO TO 12
      TFRAME = TFRAME / 2.0
      KOUNT = KOUNT + 1
      GO TO 10
   12 IF(.NOT.PT) GO TO 14
      PRINT 100, DT, AIJMAX, DTMAX,TFRAME, KOUNT
C
C
C***     WHERE M2 OR M22 CHANGED TO M3 AND M33
C
C
   14 CALL MXCLR1(M3,ID,ID)
      CALL ADDID(M3,M3,ID)
      NTERM = SIGDIG(NDIG)
      DO 15 I=1,NTERM
      FACTOR = NTERM - I + 5
      TFBYFAC = TFRAME / FACTOR
      CALL MATMUL(A,M3,M33,ID,ID,ID)
      CALL SCALR1(M33,TFBYFAC,M33,ID,ID)
   15 CALL ADDID(M33,M3,ID)
      COEFF1 = TFRAME / 12.0
      CALL SCALR1(M3,COEFF1,M33,ID,ID)
C
C
C*** THIS SECTION OF CODE ADDED TO FIND M2 FROM M3
C
C
   20 COEFF2 = TFRAME / 3.0
        CALL MATMUL(A,M33,M22,ID,ID)
        CALL ADDID(M22,M22,ID)
        CALL SCALR1(M22,COEFF2,M22,ID,ID)
        COEFF3=TFRAME/2.0
      CALL MATMUL(A,M22,M11,ID,ID,ID)
      CALL ADDID(M11,M11,ID)
      CALL SCALR1(M11,COEFF3,M11,ID,ID)
      CALL MATMUL(A,M11,M00,ID,ID,ID)
      CALL ADDID(M00,M00,ID)
      CALL SCALR1(M00,TFRAME,M00,ID,ID)
      CALL MATMUL(A,M00,MPHI,ID,ID,ID)
      CALL ADDID(MPHI,MPHI,ID)
```

```
      IF(KOUNT.EQ.0) GO TO 30
   18 KOUNT = KCUNT - 1
      TFRAME = 2.0 * TFRAME
      CALL MATMUL(MPHI,M33,M3,ID,ID,ID)
      DO 25 I=1,ID
      DO 25 J=1,ID
C
C
C***     CHANGED EXPRESSIOJN TO FIND M3
C
C
   25 M3(I,J) = 0.125 * (M3(I,J) + M33(I,J) + 3.0*M22(I,J)
     $            + 3.0 * M11(I,J) + M00(I,J))
      CALL MXEQL(M3,M33,ID,ID)
      GO TO 20
   30 CALL MXEQL(MPHI,PHIOFT,ID,ID)
      CALL MXEQL(M00,M0,ID,ID)
      CALL MXEQL(M11,M1,ID,ID)
      CALL MXEQL(M22,M2,ID,ID)
        CALL MXEQL(M33,M3,ID,ID)
      IF(.NOT.PT) GO TO 99
      PRINT 200
      CALL MATPT(A,ID,ID,0)
      PRINT 210
      CALL MATPT(PHIOFT,ID,ID,0)
      PRINT 220
      CALL MATPT(M0,ID,ID,0)
      PRINT 230
      CALL MATPT(M1,ID,ID,0)
      PRINT 240
      CALL MATPT(M2,ID,ID,0)
C
C***     ADDED TO PRINT M3
C
        PRINT 250
        CALL MATPT(M3,ID,ID,0)
   99 RETURN
  100 FORMAT(1H1,///,10X,9HDT      = ,E20.13,//,10X,
     $9HAIJMAX = ,E20.13,//,10X,9HDIMAX  = ,E20.13,
     $//,10X,9HTFRAME = ,E20.13,5X,8HBYTWC = ,I3)
  200 FORMAT(/////,14X,8HMATRIX A,/,14X,8(1H*))
  210 FORMAT(/////,14X,8HPHI OF T,/,14X,8(1H*))
  220 FORMAT(/////,14X,9HMATRIX M0,/,14X,9(1H*))
  230 FORMAT(/////,14X,9HMATRIX M1,/,14X,9(1H*))
  240 FORMAT(/////,14X,9HMATRIX M2,/,14X,9(1H*))
C
C***     ADDED
C
  250 FORMAT(/////,14X,9HMATRIX M2,/,14X,9(1H*))
      END
```

```
*       EXAMPLE TC SHCW USE OF CCMBINEC AIG.
*
*       SERVO-CONTRCLLED PENDULUM
*
 $D1
*       LINK LINEAR VARIABLES TO D1 BLOCK
*
        PROCED WAV1,WAV2,WAV3,WAV4,WAV5,
     $      WAV6,WAV7,WAV8,WAV9,WAV10=DUMMY
        CALL LINKW(WAV1,WAV2,WAV3,WAV4,WAV5,
     $      WAV6,WAV7,WAV8,WAV9,WAV10)
        ENDPRO
*
*       NONLINEAR EQUATICNS
*
        TET1.=TET2
        TET2.=-0.3*TET2+SIN(TET1)+WAV1
 $F
       SUBROUTINE GFUNC
C
C       GFUNC DEFINES THE FUNCTION G
C       FOR THE SERVO PENDULUM WHERE
C               U=G(Y,T)
C
       COMMCN/STATE1/NORDR1,Y(200),GN(200)
       COMMON/LINS/LINORD,LININP,LINCUT,A(10,10),B(10,10)
       COMMCN/LINS/C(10,10),D(10,10),F(10,10),U(10),X(10)
        U(1)=Y(1)
        U(2)=Y(2)
       RETURN
       END
        SUBRCUTINE DEFLIN
C
C       DEFLIN DEFINES THE MATRICES A,B,C,D, AND F,
C       THE INITIAL CONDITICNS CN X,
C       AND THE SIZES OF THE MATRICESAND VECTORS
C
        COMMCN/LINS/LINORD,LININP,LINOUT,A(10,10),B(10,10)
        COMMON/LINS/C(10,10),D(10,10),F(10,10),U(10),X(10)
C
C       DEFINE A,B,C,D, AND F MATRICES FOR PENDULUM
C       INITIAL CCNDITIONS FOR X ARE ZERC
C
        OMEG1=10.0
        OMEG2=1000.0
        DER=0.11
        AA=10.0
        A(1,2)=1.0
        A(2,1)=-CMEG1*OMEG2
        A(2,2)=-(CMEG1+OMEG2)
        B(2,1)=-AA*OMEG1*CMEG2
        B(2,2)=-AA*DER*OMEG1*OMEG2
        C(1,1)=1.0
        C(2,2)=1.0
        F(2,1)=1.0
C
C       DEFINE SIZES:
C               LINORD - LENGTH OF X VECTOR
C               LININP - LENGTH CF W VECTCR
C               LINOUT - LENGTH CF U VECTOR
C               LINKNO - NO. OF AVERAGES
```

```
      LINORD=2
      LININP=2
      LINOUT=2
      LINKNO=1
      RETURN
      END
 $O
*
*     INSERT LINKW, INICON, AND INTRGX HERE
*
END
    TMAX=1.8,DT=0.015,NPOINT=121,TET1=0.5
END
      L TET1,TET2
END
```

```
EXAMPLE TO SHOW USE OF GENERAL NONLINEAR ALG.

ELECTRONIC OSCILLATOR - PARTITION B


FAST SYSTEM EQUATIONS

CALL LINKW TO OBTAIN SLOW VARIABLES

  PROCED W1,W2,W3,W4,W5,W6,W7,W8,W9,W10=DUMMY
  CALL LINKW(W1,W2,W3,W4,W5,W6,W7,W8,W9,W10)
  ENDPRO
  Y1.=Y2
  Y2.=-64.0*Y1+7.5*(C+W1-Y1**2)*Y2
  X1.=1.0/R1*2.0*Y1*Y2-1.0/(R1*C1)*X1+1.0/(R1*C1)*W2


SLOW SYSTEM EQUATIONS

CALL LINKY TO OBTAIN FAST SYSTEM AVERAGES

PROCED WAV=DUMMY
CALL LINKY(WAV)
ENDPRO
  X2.=1./(R2*C1)*X1-(1./(R2*C1)+1./(R2*C2))*X2+1./(R1*C1)*WAV
  X3.=1./(R3*C2)*X2-(1./(R3*C2)+1./(R3*C3))*X3+1./(R3*C3)*X4
  X4.=1./(R4*C3)*X3-(1./(R4*C3)+1./(R4*C4))*X4+1./(R4*C4)*X5
  X5.=1./(R5*C4)*X4-(1./(R5*C4)+1./(R5*C5))*X5+1./(R5*C5)*X6
  X6.=1./(R6*C5)*X5-(1./(R6*C5)+1./(R6*C6))*X6
  V.=1.0/C6*X6
  E=FK3*(AD-FK1*0.75*SQRT(ABS(V)))
  S.=0.02*(ABS(E))**1.2*SIGN(1.0,E)-0.02*S

  SUBROUTINE LINKW(W1,W2,W3,W4,W5,W6,W7,W8,W9,W10)
  COMMON/STATE2/NORDR2,Y2(200),GN2(200)
  COMMON/SYSVAR/EXIT,RLDONE,IOUT,IFILE,IRUNNC,T,TMAX,TNEXT
  COMMON/SYSVAR/DT,DTMAX,DTMIN,EMAX,EMIN,SY(35)
  COMMON/LINK/LINKOR
  COMMON/EXTR/YEXTR(200),W(10)
*** WATCH OUT *************

  LINKOR=2
****************************
  GO TO(1,2,3,4,5,6,7,8,9,10),LINKOR
  W10=0.0
  W9=0.0
  W8=0.0
  W7=0.0
  W6=0.0
  W5=0.0
  W4=0.0
  W3=0.0
  W2=YEXTR(1)
  W1=YEXTR(7)
  RETURN
  END
SUBROUTINE LINKY(WAV)
COMMON/AVER/YAV(200)
WAV=YAV(3)
```

```
      CALL MATMUL(AVN4,B,AVN3,LINORD,LINORD,LININP)
C
C     COMPUTATION OF COEFF. MATRICES FOR LINEAR DISCRETE VALUES
C     VIZ. (AM0+I),(M0-3M1+2M2)B,
C     4(M1-M2)B,(2M2-M1)
C
      CALL MATMUL(A,M0,BETP1,LINORD,LINORD,LINORD)
      CALL ADDID(BETP1,BETP0,LINORD)
      CALL SCALR1(M1,-3.0,BETP1,LINORD,LINORD)
      CALL SCALR1(M2,2.0,BETP4,LINORD,LINORD)
      CALL MATADD(M0,BETP1,BETP1,LINORD,LINORD)
      CALL MATADD(BETP1,BETP4,BETP4,LINORD,LINORD)
      CALL MATMUL(BETP4,B,BETP1,LINORD,LINORD,LININP)
      CALL MXSUB(M1,M2,BETP4,LINORD,LINORD)
      CALL SCALR1(BETP4,4.0,BETP4,LINORD,LINORD)
      CALL MATMUL(BETP4,B,BETP2,LINORD,LINORD,LININP)
      CALL SCALR1(M2,2.0,BETP4,LINORD,LINORD)
      CALL MXSUB(BETP4,M1,BETP4,LINORD,LINORD)
      CALL MATMUL(BETP4,B,BETP3,LINORD,LINORD,LININP)
      CALL INICON
C
C
C     2. R U N   T I M E   C O M P U T A T I O N
C
C
C     COMPUTATION OF Y(N+1) AND U(N+1)
C
30    RLDONE = .FALSE.
      DO 35 J=1,NORDR1
      YSAV(J)=Y(J)
      Y(J)=Y(J)+DTO2*GN(J)
35    CONTINUE
      DO 40 J=1,LININP
      USAV(J)=U(J)
40    CONTINUE
      T=T+DTO2
      CALL GFUNC
      DO 45 J=1,LININP
      USAV2(J)=U(J)
45    CONTINUE
      CALL DIFEQ1
      DO 50 J=1,NORDR1
      Y(J)=YSAV(J)+DT*GN(J)
50    CONTINUE
      T=T+DTO2
      CALL GFUNC
C
C     COMPUTATION OF XAV(N+1)
C
      CALL MXCOL(AVN0,X,BN0,LINORD,LINORD)
      CALL MXCOL(AVN1,USAV,BN1,LINORD,LININP)
      CALL VECADD(BN0,BN1,BN2,LINORD)
      CALL MXCOL(AVN2,USAV2,BN0,LINORD,LININP)
      CALL VECADD(BN0,BN2,BN2,LINORD)
      CALL MXCOL(AVN3,U,BN0,LINORD,LININP)
      CALL VECADD(BN0,BN2,XAV,LINORD)
C
C     COMPUTATION OF WAV(N+1)
C
      DO 60 J=1,LINOUT
      WSAV(J)=WAV(J)
```

```
         RETURN
         END
 $O
*
*        INSERT RKM AND INTRGX HERE
*
END
         DT=1.0,TMAX=100.0,NPOINT=51,Y1=0.1,R1=8.0,R2=8.0,R3=8.0
          R4=8.0, C1=6.25,C2=4.25,C3=1.25,C4=1.25,DTMIN=1.0E-6
          DTMAX=0.3,R5=8.0,R6=8.0,C5=1.25,C6=1.25
          C=2.0,FK1=1.41,FK3=5.0,AD=7.2,EMAX=1.E-5,EMIN=1.E-7
END
          L Y1,X1,X2,X3,X5,V,S
END
```

APPENDIX B

SUBROUTINE MXCAL3

Subroutine MXCAL3 was derived from subroutine MXCAL (Ferguson, 1972) in order to compute matrix M3. The changes made to MXCAL are denoted by comment cards in the listing of MXCAL3 found as part of combined algorithm based on the Modified Euler Method in Appendix A. These changes are summarized here. First, the matrix M3 had to be declared as a parameter passed to the subroutine and dimensioned along with its corresponding work space matrix M33. The next several changes required only a parameter change from M2 to M3 or M22 to M33 in the computation of $M_3(T)$ (Palusinski and Wait 1978, pp. 34-46). Following this, statements had to be added to find M2 from M3 as shown here.

$$M_2 = (h/3)(AM_3 + I) \tag{C-1}$$

Finally M_3 (h) had to be calculated from M_3 (T) and printed and this required a change from the original formulation of M_2 (h).

After the changes were made, MXCAL3 was tested by comparing its matrices M_0, M_1, and M_2 with those computed by MXCAL for a fixed matrix A. The two subroutines were found to produce the same results for the three matrices thus verifying the subroutine MXCAL3.

APPENDIX C

TABLES OF PEAK ERRORS

The peak errors found for the experiments described in Chapter 5 are presented in this section in the form of tables.

Table C-1. Harmonic oscillator errors. Improved Euler Method

DT	W2	W3
0.01	5.06E-2	5.75E-2
0.02	2.02E-1	2.30E-1
0.04	8.08E-1	9.15E-1
0.08	3.25E+0	3.63E+0

Table C-2. Pendulum errors. Improved Euler without averaging

DT	CPU TIME	TET1	TET2	TAU1	TAU2
0.01	0.103	1.60E-2	9.61E-3	2.52E-2	7.77E-3
0.02	0.064	7.82E-2	5.29E-2	1.25E-1	3.21E-2
0.03	0.053	1.92E-1	1.41E-1	3.03E-1	7.48E-2
0.05	0.037	5.73E-1	4.29E-1	8.91E-1	2.23E-1
0.06	0.040	8.41E-1	6.38E-1	1.31E+0	3.30E-1
0.09	0.036	1.96E+0	1.49E+0	3.02E+0	8.67E-1
0.10	0.035	2.40E+0	1.83E+0	3.80E+0	3.46E+0

Table C-3. Pendulum errors. Improved Euler Method

DT	CPU TIME	TET1	TET2	TAU1	TAU2
0.01	0.077	5.87E-2	3.39E-2	1.56E-2	1.72E-2
0.02	0.045	5.87E-2	3.39E-2	6.06E-2	1.72E-2
0.03	0.030	2.33E-1	1.66E-1	2.39E-1	7.58E-2
0.05	0.020	5.19E-1	3.89E-1	5.33E-1	1.77E-1
0.06	0.020	1.43E+0	1.10E+0	1.45E+0	4.99E-1
0.09	0.015	2.04E+0	1.60E+0	2.08E+0	7.16E-1
0.10	0.013	4.57E+0	3.60E+0	4.59E+0	1.92E+0

Table C-4. Pendulum errors. Modified Euler Method

DT	CPU TIME	TET1	TET2	TAU1	TAU2
0.01	0.097	8.65E-2	6.44E-2	1.05E-1	2.68E-2
0.02	0.050	3.46E-1	2.77E-1	4.19E-1	1.09E-1
0.03	0.038	7.78E-1	6.35E-1	9.41E-1	2.47E-1
0.05	0.023	2.16E+0	1.78E+0	2.59E+0	6.86E-1
0.06	0.022	3.12E+0	2.58E+0	3.74E+0	1.13E+0
0.09	0.018	7.10E+0	5.90E+0	8.47E+0	5.06E+1
0.10	0.017	8.83E+0	7.28E+0	2.24E+1	2.24E+3

Table C-5. Mine-shaft errors. Improve Euler without averaging

DT	CPU TIME	OM	Y1	V2	V1	W1	W2
0.02	0.321	7.37E-3	1.36E-3	7.66E-3	1.39E-3	4.09E-4	7.40E-4
0.03	0.251	3.35E-2	3.47E-3	3.48E-2	3.54E-3	1.39E-4	1.03E-4
0.04	0.230	3.46E-2	6.27E-3	3.59E-2	6.45E-3	2.59E-3	3.56E-3
0.05	0.218	4.23E-3	8.54E-3	4.07E-3	8.77E-3	2.58E-3	4.64E-3
0.06	0.200	2.67E-2	1.25E-2	2.83E-2	1.28E-2	3.92E-3	6.80E-3
0.09	0.185	1.73E-2	2.75E-2	1.69E-2	2.83E-2	8.02E-3	1.48E-2
0.10	0.191	9.39E-2	3.82E-2	1.01E-1	3.90E-2	1.47E-2	2.12E-2
0.12	0.179	2.44E-2	4.88E-2	2.35E-2	5.01E-2	1.38E-2	2.61E-2

Table C-6. Mine-shaft errors. Improved Euler Method

DT	CPU TIME	OM	Y1	V2	V1	W1	W2
0.02	0.249	1.47E-2	1.07E-2	1.49E-2	1.08E-2	2.37E-3	4.20E-3
0.03	0.198	9.31E-3	3.91E-3	9.53E-3	3.89E-3	3.57E-3	1.67E-3
0.04	0.171	1.04E-2	1.34E-3	1.07E-2	1.31E-3	9.06E-3	5.35E-3
0.05	0.162	1.58E-2	7.70E-3	1.66E-2	7.98E-3	1.52E-2	9.80E-3
0.06	0.150	1.92E-2	1.26E-2	2.04E-2	1.29E-2	2.35E-2	1.61E-2
0.09	0.136	3.49E-2	3.27E-2	3.75E-2	3.36E-2	5.73E-2	4.15E-2
0.10	0.132	4.11E-2	4.11E-2	4.42E-2	4.22E-2	7.14E-2	5.22E-2
0.12	0.127	5.60E-2	6.16E-2	6.05E-2	6.32E-2	1.05E-1	7.75E-2

Table C-7. Oscillator (partition A) errors. Nonlinear method without averaging

DT	CPU TIME	Y1	X1	X2	X3	X5	V1
0.005	10.15	1.10E-2	4.89E-3	1.73E-3	1.02E-3	8.74E-4	8.65E-4
0.01	9.220	1.10E-2	3.05E-3	1.49E-3	1.04E-3	8.68E-4	6.84E-4
0.025	7.140	1.09E-2	1.30E-3	1.55E-3	1.08E-3	9.08E-4	7.11E-4
0.05	6.290	1.07E-2	2.12E-3	1.19E-3	8.40E-4	6.19E-4	4.54E-4
0.10	6.160	1.00E-2	9.34E-3	2.29E-3	1.60E-3	1.03E-3	6.64E-4
0.20	5.880	1.08E-2	2.19E-2	2.70E-2	2.51E-3	1.15E-3	6.23E-4

Table C-8. Oscillator (partition A) errors. Nonlinear method with averaging

DT	CPU TIME	Y1	X1	X2	X3	X5	V1
0.005	10.78	1.51E-1	2.49E-1	2.58E-2	7.94E-3	3.28E-3	3.22E-3
0.01	9.730	2.83E-1	4.66E-1	4.91E-2	1.58E-2	8.46E-3	5.30E-3
0.025	7.500	6.23E-1	1.02E+0	8.74E-2	3.91E-2	2.63E-2	2.18E-2
0.05	6.620	1.16E+0	1.91E+0	1.99E-1	7.78E-2	5.49E-2	5.04E-2
0.10	6.210	2.23E+0	3.62E+0	3.68E-1	1.55E-1	1.09E-1	1.04E-1
0.20	6.110	4.51E+0	7.20E+0	6.84E-1	2.19E-1	2.17E-1	2.08E-1

Table C-9. Oscillator (partition A) errors. Nonlinear method with shifted averaging

DT	CPU TIME	Y1	X1	X2	X3	X5	V1
0.005	25.66	3.22E-2	4.61E-2	9.22E-3	8.71E-3	8.40E-3	1.07E-2
0.01	14.57	4.53E-2	6.79E-2	1.59E-2	9.51E-3	9.62E-3	1.65E-2
0.025	10.82	8.60E-2	1.35E-1	2.16E-2	1.53E-2	1.53E-2	2.8E-2
0.05	10.01	1.19E-1	1.85E-1	2.63E-2	2.22E-2	2.21E-2	4.00E-2
0.10	9.030	1.72E-1	2.70E-1	3.11E-2	3.62E-2	3.60E-2	6.38E-2
0.20	8.760	2.89E-1	4.59E-1	5.70E-2	6.36E-2	6.16E-2	1.09E-1

Table C-10. Oscillator (partition B) errors. Nonlinear method without averaging

DT	CPU TIME	Y1	X1	X2	X3	X5	V1
0.005	9.340	1.10E-2	7.13E-3	1.08E-2	4.27E-3	2.87E-3	2.38E-3
0.01	8.260	1.97E-2	3.24E-2	4.86E-2	2.03E-2	1.34E-2	1.31E-2
0.025	6.350	1.46E-1	2.21E-1	2.45E-1	1.26E-1	9.54E-2	1.34E-1
0.05	5.430	6.06E-2	1.17E-1	3.86E-1	1.71E-1	1.22E-1	9.71E-2
0.10	5.290	9.29E-2	2.09E-1	9.74E-1	6.63E-1	4.52E-1	2.65E-1
0.20	5.180	9.01E-1	1.36E+0	2.62E+0	1.31E+0	8.52E-1	3.94E-1

Table C-11. Oscillator (partition B) errors. Nonlinear method with averaging

DT	CPU TIME	Y1	X1	X2	X3	X5	V1
0.005	10.63	1.65E-1	2.51E-1	1.50E-1	1.26E-1	1.25E-1	1.42E-1
0.01	8.890	2.40E-1	3.63E-1	2.15E-1	1.46E-1	1.52E-1	1.87E-1
0.025	6.320	1.46E-1	2.21E-1	2.45E-1	1.26E-1	9.55E-2	1.34E-1
0.05	5.760	3.39E-1	5.13E-1	3.86E-1	2.16E-1	2.17E-1	2.79E-1
0.10	5.270	6.54E-1	1.00E+0	1.06E+0	5.64E-1	4.25E-1	4.17E-1
0.20	5.020	1.12E+0	1.82E+0	1.07E+0	5.18E-1	4.79E-1	6.65E-1

Table C-12. Oscillator (partition B) errors. Nonlinear method with shifted averaging

DT	CPU TIME	Y1	X1	X2	X3	X5	V1
0.005	23.65	1.54E-2	2.11E-2	5.78E-2	5.29E-2	5.14E-2	4.76E-2
0.01	13.41	1.33E-1	1.98E-1	1.84E-1	1.54E-1	1.43E-1	1.53E-1
0.025	9.650	2.43E-1	3.87E-1	2.00E-1	1.42E-1	1.31E-1	1.80E-1
0.05	8.910	1.44E-1	1.89E-1	5.42E-1	3.66E-1	2.68E-1	2.36E-1
0.10	7.900	7.80E-2	1.80E-1	7.95E-1	3.66E-1	1.74E-1	1.04E-1
0.20	7.530	3.70E-1	7.20E-1	1.48E+0	9.20E-1	5.20E-1	2.40E-1

Table C-13. Autopilot (partition 1) errors. Nonlinear method without averaging

DT	RUN TIME	PH	X	Y	X1	X2	X3
2.5E-5	27.49	2.05E-6	6.30E-6	1.81E-6	3.33E-3	2.03E-1	4.24E-3
4.0E-5	17.09	3.15E-6	6.28E-6	1.90E-6	3.29E-3	1.80E-1	3.72E-3
5.0E-5	13.93	3.29E-6	6.32E-6	2.12E-6	3.33E-3	2.03E-1	4.24E-3
1.0E-4	7.020	1.19E-5	1.25E-5	7.54E-6	3.32E-3	2.03E-1	4.22E-3
2.0E-4	3.640	1.08E-4	1.16E-4	7.05E-5	3.21E-3	1.94E-1	4.00E-3
2.5E-4	2.910	4.10E-4	4.43E-4	2.68E-4	3.07E-3	1.70E-1	3.45E-3

Table C-14. Autopilot (partition 1) errors. Nonlinear method with averaging

DT	RUN TIME	PH	X	Y	X1	X2	X3
2.5E-5	28.29	1.26E-6	6.26E-6	1.81E-6	3.33E-3	2.03E-1	4.24E-3
4.0E-5	17.72	1.70E-6	6.20E-6	2.96E-6	3.29E-3	1.80E-1	3.72E-3
5.0E-5	14.36	4.11E-6	6.19E-6	3.51E-6	3.33E-3	2.03E-1	4.24E-3
1.0E-4	7.190	1.24E-5	1.14E-5	1.55E-5	3.32E-3	2.03E-1	4.22E-3
2.0E-5	3.700	1.01E-4	9.58E-5	1.49E-4	3.19E-3	1.94E-1	4.00E-3
2.5E-5	3.100	9.97E-5	2.04E-4	4.04E-4	2.99E-3	1.70E-1	3.65E-3

Table C-15. Autopilot (partition 1) errors. Nonlinear method with shifted averaging

DT	RUN TIME	PH	X	Y	X1	X2	X3
2.5E-5	70.25	8.87E-4	8.62E-4	5.56E-4	3.42E-3	2.03E-1	4.24E-3
4.0E-5	43.01	1.42E-3	1.38E-3	8.89E-4	3.43E-3	1.80E-1	3.73E-3
5.0E-5	34.38	1.77E-3	1.72E-3	1.11E-3	3.51E-3	2.03E-1	4.25E-3
1.0E-4	17.58	3.55E-3	3.47E-3	2.22E-3	3.69E-3	2.03E-1	4.26E-3
2.0E-4	8.690	7.16E-3	6.96E-3	4.42E-3	4.05E-3	2.02E-1	4.28E-3
2.5E-5	7.120	9.21E-3	8.99E-3	5.47E-3	4.28E-3	1.91E-1	4.06E-3

Table C-16. Autopilot (partition 2) errors. Nonlinear method without averaging

DT	RUN TIME	PH	X	Y	X1	X2	X3
2.5E-5	26.97	2.01E-6	6.36E-6	1.88E-6	3.33E-3	2.04E-1	4.25E-3
4.0E-5	16.87	2.45E-6	6.36E-6	1.88E-6	3.31E-3	1.81E-1	3.76E-3
5.0E-5	13.82	1.94E-6	6.36E-6	1.88E-6	3.36E-3	2.05E-1	4.30E-3
1.0E-4	6.990	1.82E-6	6.36E-6	1.88E-6	3.42E-3	2.10E-1	4.47E-3
2.0E-4	3.610	1.51E-6	6.31E-6	1.73E-6	3.87E-3	2.21E-1	4.94E-3
2.5E-4	2.960	3.61E-6	6.17E-6	2.62E-6	4.01E-3	2.13E-1	4.01E-3

Table C-17. Autopilot (partition 2) errors. Nonlinear method with averaging

DT	RUN TIME	PH	X	Y	X1	X2	X3
2.5E-5	27.39	1.65E-5	6.36E-5	1.15E-5	3.33E-3	2.04E-1	4.25E-3
4.0E-5	17.05	4.14E-5	3.45E-5	3.09E-5	3.31E-3	1.81E-1	3.76E-3
5.0E-5	13.84	6.25E-5	5.31E-5	4.72E-5	3.35E-3	2.05E-1	4.30E-3
1.0E-4	6.910	2.44E-4	2.14E-4	1.88E-4	3.41E-3	2.07E-1	4.48E-3
2.0E-4	3.570	5.25E-4	5.08E-4	3.23E-4	3.66E-3	2.04E-1	4.93E-3
2.5E-5	3.100	1.86E-3	2.57E-3	1.87E-3	5.09E-3	1.90E-1	4.14E-3

Table C-18. Autopilot (partition 2) errors. Nonlinear method with shifted averaging

DT	RUN TIME	PH	X	Y	X1	X2	X3
2.5E-5	68.62	2.79E-5	2.46E-5	1.53E-5	3.71E-3	2.15E-1	4.26E-3
4.0E-5	42.87	4.19E-5	3.82E-5	2.38E-5	3.91E-3	1.98E-1	3.76E-3
5.0E-5	34.42	5.22E-5	4.80E-5	2.98E-5	4.11E-3	2.26E-1	4.29E-3
1.0E-4	17.44	9.01E-5	8.94E-5	5.55E-5	5.27E-3	2.51E-1	4.40E-3
2.0E-4	8.740	1.34E-4	1.52E-4	9.60E-5	9.49E-3	3.03E-1	4.75E-3
2.5E-5	7.090	1.45E-4	2.00E-4	1.10E-4	1.15E-2	3.19E-1	4.77E-3

APPENDIX D

COMBINED ALGORITHM WITHOUT AVERAGING

The combined algorithm without averaging (labelled CRK2HI.C) that was compared with the combined algorithms with averaging is described in Palusinski, 1977b, pp. 21-23. This algorithm performs the half step calculations

$$k_1 = f\,(y_n, t_n, w_n) \tag{D-1}$$

$$y_{n+1/2} = y_n + (h/2)k_1 \tag{D-2}$$

$$u_{n+1/2} = g\,(y_{n+1/2}, y_{n+1/2}) \tag{D-3}$$

The nonlinear full step solution is then predicted as

$$y^p_{n+1} = y_n + hk_1 \tag{D-4}$$

$$u^p_{n+1} = g\,(y^p_{n+1}, t_{n+1}) \tag{D-5}$$

The linear system based on the predicted u^p_{n+1} value is given by

$$\begin{aligned} x^p_{n+1} &= e^{Ah}x_n + (M_o - 3M_1 + 2M_2)Bu_n \\ &\quad + 4(M_1 - M_2)B_{u_{n+1/2}} + (2m_2 - M_1)Bu^p_{n+1} \end{aligned} \tag{D-6}$$

$$w^p_{n+1} = Cx^p_{n+1} + Du^p_{n+1} \tag{D-7}$$

The nonlinear full step solution is obtained from

$$k_2 = f\,(y_n+hk_1,\ t_{n+1},\ w_{n+1}^p) \tag{D-8}$$

$$y_{n+1} = y_n + (h/2)(k_1+k_2) \tag{D-9}$$

$$u_{n+1} = g\,(y_{n+1},\ t_{n+1}) \tag{D-10}$$

Finally, the nonlinear correction is computed

$$\begin{aligned} x_{n+1} &= e^{Ah}x_m + (M_o-3M_1+2M_2)Bu_n+4(M_1-M_2) \\ &\quad + 2(m_2-M_1)\ Bu_{n+1} \end{aligned} \tag{D-11}$$

$$w_{n+1} = Cx_{n+1} + Du_{n+1} \tag{D-12}$$

As seen, this algorithm is more complicated than the Improved Euler algorithm since half step solutions are required.

LIST OF REFERENCES

Ferguson, R. E. A Matrix-Vector Manipulation Software Package. Tucson: Department of Electrical Engineering CSRL Memo 231, 1972.

Gille, J. C., Pelegrin, M. J., and Decaulne, P. Feedback Control Systems. New York: McGraw Hill Book Co., 1959.

Korn, G. A. and Korn, T. M. Electronic Analog Computers. New York: McGraw Hill Book Co., 1956.

Korn, G. A. and Wait, J. V. Digital Continuous-System Simulation. Englewood Hills, N. J.: Prentice Hall, 1977.

Lucas, J. J. and Wait, J. V. DAREP--a portable CSSL-type simulation language, Simulation, January 1975, 24, 17-28.

Palusinski, O. A. Averaging in Simulation of Coupled Linear and Nonlinear Dynamic Systems. Tucson: Department of Electrical Engineering CSRL Memo 312, 1977a.

Palusinski, O. A. Multirate Integration Routines for Partitioned Dynamic Systems. Tucson: Department of Electrical Engineering CSRL Memo 314, 1977b.

Palusinski, O. A. Averaging in Simulation of Partitioned Dynamic Systems. Tucson: Department of Electrical Engineering, 1978.

Palusinski, O. A., Skowronek, M., and Znamirowski, L. Analog and Hybrid Simulation. Warsaw: WNT, 1976.

Palusinski, O. A., and Wait, J. V. Simulation Methods for Combined Linear and Nonlinear Systems, Simulation, March 1978, 30, 85-94.