

CANADIAN THESES ON MICROFICHE

I.S.B.N.

THESES CANADIENNES SUR MICROFICHE



National Library of Canada
Collections Development Branch

Canadian Theses on
Microfiche Service

Ottawa, Canada
K1A 0N4

Bibliothèque nationale du Canada
Direction du développement des collections

Service des thèses canadiennes
sur microfiche

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us a poor photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de mauvaise qualité.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS RECUE



National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Division

Division des thèses canadiennes

Ottawa, Canada
K1A 0N4

60324

PERMISSION TO MICROFILM — AUTORISATION DE MICROFILMER

- Please print or type — Écrire en lettres moulées ou dactylographier

Full Name of Author — Nom complet de l'auteur

BERNHARD MICHAEL JATZECK

Date of Birth — Date de naissance

55-10-12

Country of Birth — Lieu de naissance

FEDERAL REPUBLIC OF GERMANY

Permanent Address — Résidence fixe

#305 - 115 RBINDEER RD.
SASKATOON, SASK.
SK 4T8

Title of Thesis — Titre de la thèse

NUMERICAL SOLUTION OF SOME SECOND-ORDER
DIFFERENTIAL EQUATIONS

University — Université

UNIVERSITY OF ALBERTA.

Degree for which thesis was presented — Grade pour lequel cette thèse fut présentée

M.Sc.

Year this degree conferred — Année d'obtention de ce grade

1982

Name of Supervisor — Nom du directeur de thèse

DR. J. B. HADDOW

Permission is hereby granted to the NATIONAL LIBRARY OF CANADA to microfilm this thesis and to lend or sell copies of the film.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

L'autorisation est, par la présente, accordée à la BIBLIOTHÈQUE NATIONALE DU CANADA de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans l'autorisation écrite de l'auteur.

Date

02-08-06

Signature

Bernhard Michael Jatzeck

THE UNIVERSITY OF ALBERTA

NUMERICAL SOLUTION OF SOME
SECOND-ORDER DIFFERENTIAL EQUATIONS

by

BERNHARD MICHAEL JATZECK, B. SC., MEC. E. (ALTA.)

* A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE
OF MASTER OF SCIENCE

MECHANICAL ENGINEERING

EDMONTON, ALBERTA

FALL 1982

THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR BERNHARD MICHAEL JATZECK, B. SC., MEC. E. (ALTA.)

TITLE OF THESIS NUMERICAL SOLUTION OF SOME SECOND-ORDER DIFFERENTIAL
EQUATIONS

DEGREE FOR WHICH THESIS WAS PRESENTED MASTER OF SCIENCE

YEAR THIS DEGREE GRANTED FALL 1982

Permission is hereby granted to THE UNIVERSITY OF ALBERTA LIBRARY to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

(SIGNED)

Bernhard Michael Jatzeck

PERMANENT ADDRESS:

... #305 ... 15 REINDEER RD ...

... SASKATOON, SASK ...

... S7K 4W8 ...

DATED ... 820806 19 ...

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled NUMERICAL SOLUTION OF SOME SECOND-ORDER DIFFERENTIAL EQUATIONS submitted by BERNHARD MICHAEL JATZECK, B. SC., MEC. E. (ALTA.) in partial fulfilment of the requirements for the degree of MASTER OF SCIENCE.

J. B. Kaddour.....
Supervisor

M. M. M.
R. J. Danl.....

Date .. 20th July 1972.....

ABSTRACT

The solution of some second-order differential equations using the method of piecewise linearization is examined. These are the undamped and damped hard spring equations (linear and nonlinear damping) and the Van der Pol and Mathieu equations.

Two methods of piecewise linearization are considered: solution by chords and solution by tangents. The two sets of results that are obtained are compared to the approximation to the exact solution of the undamped hard spring equation. The former method of linearization is chosen as the solution to the remainder of the equations that are examined.

Generally, the piecewise linearization solution gives a good agreement when compared to either the Runge-Kutta solution or the approximation to the exact solution (if available). Exceptions to this appear in some of the results for the Van der Pol and Mathieu equations.

The programmed piecewise linear methods are generally slower in execution time than either the Runge-Kutta solution or the approximation to the exact solution. The exceptions to this are the undamped hard spring equation and some instances for the Mathieu equation.

ACKNOWLEDGEMENTS

Several individuals contributed to the completion of this thesis.

The following names I mention are among those whose efforts are deserving of my special thanks.

My supervisor, Dr. J. B. Haddow, was most helpful in the research and the writing. His guidance and advice in the material that was examined and the methods of solution that were used is greatly appreciated. I also wish to thank the other members of the examining committee for their efforts.

The personnel of Telsec Business Centre, Saskatoon, particularly Mrs. Carole Fontaine, were most patient in the preparation of the manuscript. I also wish to thank Mrs. Faye Magera, also of Saskatoon, for her initial efforts at this task.

The computer programming was a major endeavour and the advice of fellow graduate student Mr. Roger Toogood was a valuable asset. The advice of the consultants at the University of Alberta Computing Services, particularly Dr. Ron Torgerson, was also appreciated.

I am grateful to those department faculty and staff members for their efforts which allowed the completion of my degree. In particular, I thank Dr. D. G. Bellow, department chairman, for making it all possible.

A brief word of thanks also goes to some fellow students and some of the staff at the University of British Columbia, Department of Mechanical Engineering, from whom I learned much while I was there.

Finally, I wish to thank the many authors and/or publishers who were kind to allow me to use or cite material from their works.

TABLE OF CONTENTS

<u>Chapter</u>	<u>Page</u>
ABSTRACT	iv
ACKNOWLEDGEMENTS	v
LIST OF TABLES	viii
LIST OF FIGURES.....	ix
LIST OF SYMBOLS	xv
I. INTRODUCTION	1
II. HARD SPRING EQUATION (UNDAMPED)	5
A. Preliminary Comments	5
B. Approximation to Exact Solution	5
C. Derivation of Piecewise Linear Solution by Chords	9
D. Derivation of Piecewise Linear Solution by Tangents	16
E. Results	23
III. HARD SPRING EQUATION (LINEAR DAMPING)	35
A. Preliminary Comments	35
B. Derivation of Piecewise Linear Solution	37
C. Runge-Kutta Solution	39
D. Results	39
IV. HARD SPRING EQUATION (NONLINEAR DAMPING).....	58
A. Preliminary Comments	58
B. Derivation of Piecewise Linear Solution	58
C. Results	62
V. VAN DER POL EQUATION	79
A. Preliminary Comments	79
B. Derivation of Piecewise Linear Solution	80
C. Results	87

TABLE OF CONTENTS (continued)

<u>Chapter</u>	<u>Page</u>
VI. MATHIEU EQUATION	135
A. Preliminary Comments	135
B. Generation of Characteristic Number	135
C. Series Approximation to Exact Solution	142
D. Derivation of Piecewise Linear Solution	147
E. Results	150
VII. DISCUSSION	184
VIII. CONCLUSIONS	197
IX. RECOMMENDATIONS FOR FURTHER WORK	199
A. Mathieu's Equation	199
B. Two-Degree-of-Freedom Nonlinear Equations	200
REFERENCES	201
APPENDIX	206
A. HSPRING	207
B. HSPRINGC	214
C. HSPRINGQ	221
D. VANDERPOL	228
E. MATHIEU	237
VITA	247

LIST OF TABLES

<u>Table</u>		<u>Page</u>
2.1	Undamped Hard Spring Equation Periods for $a = 1.0, b = 0.15$	24
2.2	Undamped Hard Spring Equation Periods for $a = 1.0, b = 2.0$	26

LIST OF FIGURES

<u>Figure</u>		<u>Page</u>
1.	Methods of Piecewise Linearization	4
2.1	Mass-Spring System (Undamped)	6
2.2	Approximation by Chords	9
2.3	Chord Segment Construction	12
2.4	Approximation by Tangents	17
2.5	Tangent Segment Construction	18
2.6	Solutions to Hard Spring Equation (Undamped) for $a = 1.0$, $b = 0.15$, $\Delta t = \pi/4/100.0$, $\Delta x = 0.20$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	27
2.7	Solutions to Hard Spring Equation (Undamped) for $a = 1.0$, $b = 0.15$, $\Delta t = \pi/4/100.0$, $\Delta x = 0.10$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	28
2.8	Solutions to Hard Spring Equation (Undamped) for $a = 1.0$, $b = 2.0$, $\Delta t = \pi/4/100.0$, $\Delta x = 0.20$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	29
2.9	Solutions to Hard Spring Equation (Undamped) for $a = 1.0$, $b = 2.0$, $\Delta t = \pi/4/100.0$, $\Delta x = 0.10$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	30
2.10	Execution Times for Solutions to Hard Spring Equation (Undamped) for $a = 1.0$, $b = 0.15$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	32
2.11	Execution Times for Solutions to Hard Spring Equation (Undamped) for $a = 1.0$, $b = 2.0$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	33
3.1	Mass-Spring System (Linear Damping)	36
3.2	Solutions to Hard Spring Equation (Linear Damping) for $a = 1.0$, $b = 0.15$, $\frac{C}{M} = 0.0$, $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	41
3.3	Solutions to Hard Spring Equation (Linear Damping) for $a = 1.0$, $b = 2.0$, $\frac{C}{M} = 0.0$, $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	43
3.4	Solutions to Hard Spring Equation (Linear Damping) for $a = 1.0$, $b = 0.15$, $\frac{C}{M} = 0.10$, $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	45

LIST OF FIGURES (cont'd)

<u>Figure</u>	<u>Page</u>
3.5 Solutions to Hard Spring Equation (Linear Damping) for $a = 1.0$, $b = 2.0$, $\frac{C}{M} = 0.10$, $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	47
3.6 Solutions to Hard Spring Equation (Linear Damping) for $a = 1.0$, $b = 0.15$, $\frac{C}{M} = 0.25$, $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	49
3.7 Solutions to Hard Spring Equation (Linear Damping) for $a = 1.0$, $b = 2.0$, $\frac{C}{M} = 0.25$, $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	51
3.8 Execution Times for Solutions to Hard Spring Equation (Linear Damping) for $a = 1.0$, $\frac{C}{M} = 0.0$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	53
3.9 Execution Times for Solutions to Hard Spring Equation (Linear Damping) for $a = 1.0$, $\frac{C}{M} = 0.10$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	54
3.10 Execution Times for Solutions to Hard Spring Equation (Linear Damping) for $a = 1.0$, $\frac{C}{M} = 0.25$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	55
4.1 Mass-Spring System (Nonlinear Damping)	59
4.2 Solutions to Hard Spring Equation (Nonlinear Damping) for $a = 1.0$, $b = 0.15$, $\frac{q}{M} = 0.0$, $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	64
4.3 Solutions to Hard Spring Equation (Nonlinear Damping) for $a = 1.0$, $b = 2.0$, $\frac{q}{M} = 0.0$, $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	66
4.4 Solutions to Hard Spring Equation (Nonlinear Damping) for $a = 1.0$, $b = 0.15$, $\frac{q}{M} = 0.10$, $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	68
4.5 Solutions to Hard Spring Equation (Nonlinear Damping) for $a = 1.0$, $b = 2.0$, $\frac{q}{M} = 0.10$, $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	70
4.6 Solutions to Hard Spring Equation (Nonlinear Damping) for $a = 1.0$, $b = 0.15$, $\frac{q}{M} = 0.25$, $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	72

LIST OF FIGURES (cont'd)

<u>Figure</u>		<u>Page</u>
4.7	Solutions to Hard Spring Equation (Nonlinear Damping) for $a = 1.0$, $b = 2.0$, $\frac{q}{M} = 0.25$, $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	74
4.8	Execution Times for Solutions to Hard Spring Equation (Nonlinear Damping) for $a = 1.0$, $\frac{q}{M} = 0.0$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	75
4.9	Execution Times for Solutions to Hard Spring Equation (Nonlinear Damping) for $a = 1.0$, $\frac{q}{M} = 0.10$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	76
4.10	Execution Times for Solutions to Hard Spring Equation (Nonlinear Damping) for $a = 1.0$, $\frac{q}{M} = 0.25$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	77
5.1	Typical Interval Layout (Van der Pol Equation)	82
5.2	Layout of Peak or Trough (Van der Pol Equation)	84
5.3	Solutions to Van der Pol Equation for $\mu = 0.25$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	89
5.4	Phase Trajectories for Van der Pol Equation for $\mu = 0.25$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	91
5.5	Solutions to Van der Pol Equation for $\mu = 0.25$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.0$, $\dot{x}(0) = 0.0$	93
5.6	Phase Trajectories for Van der Pol Equation for $\mu = 0.25$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.0$, $\dot{x}(0) = 0.0$	95
5.7	Solutions to Van der Pol Equation for $\mu = 0.25$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.5$, $\dot{x}(0) = 0.0$	97
5.8	Phase Trajectories for Van der Pol Equation for $\mu = 0.25$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.5$, $\dot{x}(0) = 0.0$	99
5.9	Solutions to Van der Pol Equation for $\mu = 1.0$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.5$, $\dot{x}(0) = 0.0$	102
5.10	Phase Trajectories for Van der Pol Equation for $\mu = 1.0$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.5$, $\dot{x}(0) = 0.0$	104

LIST OF FIGURES (cont'd)

<u>Figure</u>		<u>Page</u>
5.11	Solutions to Van der Pol Equation for $\mu = 2.0$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.5$, $\dot{x}(0) = 0.0$	106
5.12	Phase Trajectories for Van der Pol Equation for $\mu = 2.0$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.5$, $\dot{x}(0) = 0.0$	108
5.13	Solutions to Van der Pol Equation for $\mu = 5.0$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	110
5.14	Phase Trajectories for Van der Pol Equation for $\mu = 5.0$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	112
5.15	Solutions to Van der Pol Equation for $\mu = 5.0$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.0$, $\dot{x}(0) = 0.0$	114
5.16	Phase Trajectories for Van der Pol Equation for $\mu = 5.0$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.0$, $\dot{x}(0) = 0.0$	116
5.17	Solutions to Van der Pol Equation for $\mu = 5.0$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.5$, $\dot{x}(0) = 0.0$	118
5.18	Phase Trajectories for Van der Pol Equation for $\mu = 5.0$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.5$, $\dot{x}(0) = 0.0$	120
5.19	Solutions to Van der Pol Equation for $\mu = 5.0$, $\Delta t_p = 0.05$, $\Delta t_{RK} = 0.05$, $x(0) = 2.5$, $\dot{x}(0) = 0.0$	122
5.20	Phase Trajectories for Van der Pol Equation for $\mu = 5.0$, $\Delta t_p = 0.05$, $\Delta t_{RK} = 0.05$, $x(0) = 2.5$, $\dot{x}(0) = 0.0$	124
5.21	Solutions to Van der Pol Equation for $\mu = 5.0$, $\Delta t_p = 0.25$, $\Delta t_{RK} = 0.25$, $x(0) = 2.5$, $\dot{x}(0) = 0.0$	126
5.22	Phase Trajectories for Van der Pol Equation for $\mu = 5.0$, $\Delta t_p = 0.25$, $\Delta t_{RK} = 0.25$, $x(0) = 2.5$, $\dot{x}(0) = 0.0$	128
5.23	Execution Times for Van der Pol Equation Solutions for $\mu = 0.25$, $x(0) = 3.0$, $\dot{x}(0) = 0.0$	131
5.24	Execution Times for Van der Pol Equation Solutions for $\mu = 1.0$, $x(0) = 3.0$, $\dot{x}(0) = 0.0$	132
5.25	Execution Times for Van der Pol Equation Solutions for $\mu = 2.0$, $x(0) = 3.0$, $\dot{x}(0) = 0.0$	133

LIST OF FIGURES (cont'd)

<u>Figure</u>		<u>Page</u>
5.26	Execution Times for Van der Pol Equation Solutions for $\mu = 5.0$, $x(0) = 3.0$, $\dot{x}(0) = 0.0$	134
6.1	Typical Interval Layout (Mathieu Equation)	149
6.2	Solutions to Mathieu Equation for $a_{GUESS} = -50.0$, $q = 1.0$, $\Delta z = \pi/200.0$, $y(0) = 0.0$, $y'(0) = 1.0$	153
6.3	Solutions to Mathieu Equation for $a_{GUESS} = 10.0$, $q = 1.0$, $\Delta z = \pi/200.0$, $y(0) = 1.0$, $y'(0) = 0.0$	155
6.4	Solutions to Mathieu Equation for $a_{GUESS} = 30.0$, $q = 1.0$, $\Delta z = \pi/200.0$, $y(0) = 0.0$, $y'(0) = 1.0$	157
6.5	Solutions to Mathieu Equation for $a_{GUESS} = 10.0$, $q = 10.0$, $\Delta z = \pi/200.0$, $y(0) = 0.0$, $y'(0) = 1.0$	159
6.6	Solutions to Mathieu Equation for $a_{GUESS} = 30.0$, $q = 10.0$, $\Delta z = \pi/200.0$, $y(0) = 0.0$, $y'(0) = 1.0$	161
6.7	Solutions to Mathieu Equation for $a_{GUESS} = -50.0$, $q = 10.0$, $\Delta z = \pi/200.0$, $y(0) = 0.0$, $y'(0) = 1.0$	163
6.8	Solutions to Mathieu Equation for $a_{GUESS} = 10.0$, $q = 20.0$, $\Delta z = \pi/200.0$, $y(0) = 1.0$, $y'(0) = 0.0$	165
6.9	Solutions to Mathieu Equation for $a_{GUESS} = -50.0$, $q = 20.0$, $\Delta z = \pi/200.0$, $y(0) = 0.0$, $y'(0) = 1.0$	167
6.10	Solutions to Mathieu Equation for $a_{GUESS} = 30.0$, $q = 30.0$, $\Delta z = \pi/200.0$, $y(0) = 0.0$, $y'(0) = 1.0$	169
6.11	Solutions to Mathieu Equation for $a_{GUESS} = -50.0$, $q = 30.0$, $\Delta z = \pi/200.0$, $y(0) = 0.0$, $y'(0) = 1.0$	171
6.12	Solutions to Mathieu Equation for $a_{GUESS} = 30.0$, $q = 40.0$, $\Delta z = \pi/200.0$, $y(0) = 1.0$, $y'(0) = 0.0$	173
6.13	Solutions to Mathieu Equation for $a_{GUESS} = -50.0$, $q = 40.0$, $\Delta z = \pi/200.0$, $y(0) = 1.0$, $y'(0) = 0.0$	175
6.14	Solutions to Mathieu Equation for $a_{GUESS} = -50.0$, $q = 40.0$, $\Delta z = \pi/400.0$, $y(0) = 1.0$, $y'(0) = 0.0$	178
6.15	Execution Times for Solutions to Mathieu Equation for $a_{GUESS} = 10.0$, $q = 1.0$, $y(0) = 1.0$, $y'(0) = 0.0$	180

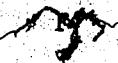
LIST OF FIGURES (cont'd)

<u>Figure</u>		<u>Page</u>
6.16	Execution Times for Solutions to Mathieu Equation for $a_{GUESS} = 30.0, q = 20.0, y(0) = 1.0, y'(0) = 0.0$	181
6.17	Execution Times for Solutions to Mathieu Equation for $a_{GUESS} = -50.0, q = 40.0, y(0) = 1.0, y'(0) = 0.0$	182
7.1	Solutions to Mathieu Equation for $a_{GUESS} = -50.0,$ $q = 20.0, \Delta z = \pi/200.0, y(0) = 0.0, y'(0) = 1.0$	193
7.2	Stability Chart for Piecewise Linear Solution to Mathieu Equation	194

LIST OF SYMBOLS

a, b	Hard spring restoring force coefficients
a'	Characteristic number (general use and for non-zero initial displacement and zero initial velocity)
a_{GUESS}	Initial guess at characteristic number
a_{IN}	Input value for characteristic number continued fraction (scan phase)
a_{OUT}	Output for preceding continued fraction (scan phase)
a_{MID}	Input value for characteristic number continued fraction (interval halving search)
a_{MIDOUT}	Output value for preceding continued fraction
$a_{\text{UP}}, a_{\text{LOW}}$	Upper and lower bounds, respectively (scan phase)
a_{2n}	Characteristic number (even order, non-zero initial displacement, and zero initial velocity)
A_{2r}	Coefficient for series approximation to Mathieu function (even order and preceding initial conditions)
a_{2n+1}	Characteristic number (odd order and preceding initial conditions)
A_{2r+1}	Coefficient for series approximation to Mathieu function (odd order and preceding initial conditions)
b_{2n+1}	Characteristic number (odd order, zero initial displacement and non-zero initial velocity)
B_{2n+1}	Coefficient for series approximation to Mathieu function (odd order and preceding initial conditions)
b_{2n+2}	Characteristic number (even order and preceding initial conditions)
B_{2r+2}	Coefficient for series approximation to Mathieu function (even order and preceding initial conditions)
A, B	General equation coefficients for linear segment solution (trigonometric)
c, p	Damping constant and angular frequency, respectively, for linear approximation
c	Damping constant

LIST OF SYMBOLS (cont'd)

C, D	General equation coefficients for $\ddot{x}[(\Delta t)_j]$ (hyperbolic)
$f(a, q, n)$	General term for characteristic number continued fraction
$f(x, \dot{x}, t)$	Generalized term representing damping and restoring forces
F	Generalized restoring force
F_0	Restoring force at beginning of segment (method of chords)
F_1	Restoring force at end of segment (method of chords)
F'_0	Slope of tangent line at x_0 (method of tangents)
	Slope of tangent line at x_1 (method of tangents)
$F(\lambda, \phi)$	Incomplete elliptic integral of first kind
k_0	Slope of linear segment (method of chords)
$K(\lambda^2)$	Complete elliptic integral of first kind
m	Number of linear segments per quarter-cycle (piecewise linear solution)
M	System mass
n	Number of increments per quarter-cycle (approximation to exact solution of undamped hard spring equation)
n	$c/2M$
p	Trigonometric angular frequency for linear segment (piecewise linear solution)
p^*	Damped natural frequency
p^{**}	$\sqrt{n^2 - p^2}$ (Van der Pol equation)
p_m	Trigonometric angular frequency for m th segment
q	Nonlinear damping constant (hard spring equation, nonlinear damping)
q	Parameter for a (Mathieu equation)
s	ip

LIST OF SYMBOLS (cont'd)

s_m	Hyperbolic angular frequency for mth segment (Mathieu equation)
t	Time
t_0	Cumulative time at beginning of linear segment (method of chords)
t_1	Cumulative time at end of linear segment (method of chords)
$(t_1 - t_0)_i$	Interim value for $t_1 - t_0$ for Newton-Raphson relations (damped hard spring equation)
t_{CO}	Cumulative time at beginning of linear segment (method of tangents)
t_{Cl}	Cumulative time at end of linear segment (method of tangents)
x	Displacement
$x[(t_1 - t_0)_i]$	Displacement using $(t_1 - t_0)_i$ (damped hard spring equation)
\dot{x}	Velocity
$\dot{x}[(t_1 - t_0)_i]$	Velocity using $(t_1 - t_0)_i$ (damped hard spring equation)
$\dot{x}[(\Delta t)_i]$	Velocity using $(\Delta t)_i$
\ddot{x}	Acceleration
$\ddot{x}[(t_1 - t_0)_i]$	Acceleration using $(t_1 - t_0)_i$ (damped hard spring equation)
$\ddot{x}[(\Delta t)_i]$	Acceleration using $(\Delta t)_i$
x_0	Displacement at beginning of linear segment
x_1	Displacement at end of linear segment
\dot{x}_0	Velocity at beginning of linear segment (method of chords)
\dot{x}_1	Velocity at end of linear segment (method of chords)
x_0^*	Relative equilibrium point for linear segment (method of chords)

LIST OF SYMBOLS (cont'd)

x_F	Estimated value for x_{11}
x_{PREV}	Previous value of estimated linear segment displacement
x_{C0}	Crossing point tangents from x_0 to x_1 for (method of tangents)
x_{C1}	Crossing point for tangents from x_1 and following linear segment (method of tangents)
x_{11}	Displacement at peak or trough
x_{12}	Displacement at end of linear segment in which peak or trough occurs
\dot{x}_{AV}	Average segment velocity (hard spring equation, nonlinear damping)
\dot{x}_{C0}	Velocity at x_{C0} (method of tangents)
\dot{x}_{C1}	Velocity at x_{C1} (method of tangents)
x_0	Initial system displacement
y	Displacement (Mathieu equation)
y'	Velocity (Mathieu equation)
y	Acceleration (Mathieu equation)
y_{m-1}, y_m	Displacements at end of $(m - 1)$ th and m th segments, respectively
y'_{m-1}, y'_m	Velocities at end of $(m - 1)$ th and m th segments, respectively
y_0	Displacement at beginning of segment
y'_0	Velocity at beginning of segment
X_0	Initial system displacement (Mathieu equation)
Y'_0	Initial velocity (Mathieu equation)
z	Independent variable (Mathieu equation)
α, β	Coefficients for approximation to exact solution to Mathieu equation
α, β, γ	Trigonometric equation coefficients for finding $t_1 - t_0$ (undamped hard spring equation)

LIST OF SYMBOLS (cont'd)

α_v	Generalized numerator for continued fraction
β_v	Generalized denominator for continued fraction
γ_{n+1}, γ_n	Sum of characteristic number continued fraction terms prior to $(n + 1)$ th or n th terms
Δt	Time increment (approximation to exact solution to undamped hard spring equation)
Δt_p	Time interval for piecewise linear solution (Van der Pol equation)
Δt_{RK}	Time interval for Runge-Kutta solution (Van der Pol equation)
Δt_1	Interval from t_0 to time at which peak or trough occurs
Δt_2	Interval from peak or trough to t_1
$(\Delta t)_1$	Interim value of Δt_1 as used in Newton-Raphson method
Δx	Displacement increment (piecewise linear solution)
Δz	Increment of z
θ	Solution to preceding trigonometric equation
λ^2	Parameter for $F(\lambda, \phi)$
μ	Equation parameter (Van der Pol equation)
v	Term number for continued fraction
τ	Period of oscillation (Mathieu equation)
τ_π	Period of oscillation (approximation to exact solution of undamped hard spring equation)
$\tau_\pi/4$	Quarter-period of oscillation (approximation to exact solution of undamped hard spring equation)
ϕ	Amplitude for $F(\lambda, \phi)$
$\phi(r)$	Order of approximation to exact solution of Mathieu equation
ϕ_{n+1}	Denominator for characteristic number continued fraction

LIST OF SYMBOLS (cont'd)

ω

Angular frequency of oscillation for Jacobian elliptic function

I. INTRODUCTION

Oscillating systems are common in mechanical and electrical engineering, as well as in other areas of science such as chemistry and biology. These can often be described by using differential equations, either nonlinear or linear with time-varying coefficients.

Exact solutions for these equations are often difficult to obtain (if at all), but reasonably good approximations can usually be achieved using a variety of methods. Some of the better-known ones are variation of parameters [1], reversion [2], and perturbation [3].

One method that is mentioned in some textbooks but is often not given as much prominence as the others is that of piecewise linearization, the approximation of an equation by one that is linear. This approach is investigated in this thesis, applying it to a number of second-order differential equations, comparing the results with either an approximation to an exact solution (if available), such as [4], or to a solution obtained from a Runge-Kutta method [5].

What this method does is to approximate the equation being examined:

$$\ddot{x} + f(x, \dot{x}, t) = 0 , \quad (1.1)$$

by one that is linear. The terms corresponding to the damping and restoring forces in (1.1) are represented by $f(x, \dot{x}, t)$. The linear equation is that for a mass-spring oscillator:

$$\begin{aligned} \ddot{x} + \frac{c}{M} \dot{x} + \frac{k}{M} &= 0 \\ &= \ddot{x} + 2nx + p^2 x , \end{aligned} \quad \left. \right\} \quad (1.2)$$

where:

c = Coefficient of damping,

M = System mass,

k = Spring constant,

p = Undamped natural frequency.

The key is to approximate the coefficients of x and \dot{x} in (1.1) by constants that would correspond to n and p in (1.2).

This is done by considering the restoring force-displacement curve of (1.1). For a small section of this curve, a straight line could be used as an approximation to it over the same region in which this section lies [6]. The curve can be thought of consisting of a sequence of these small sections, and each will have its own unique linear approximation.

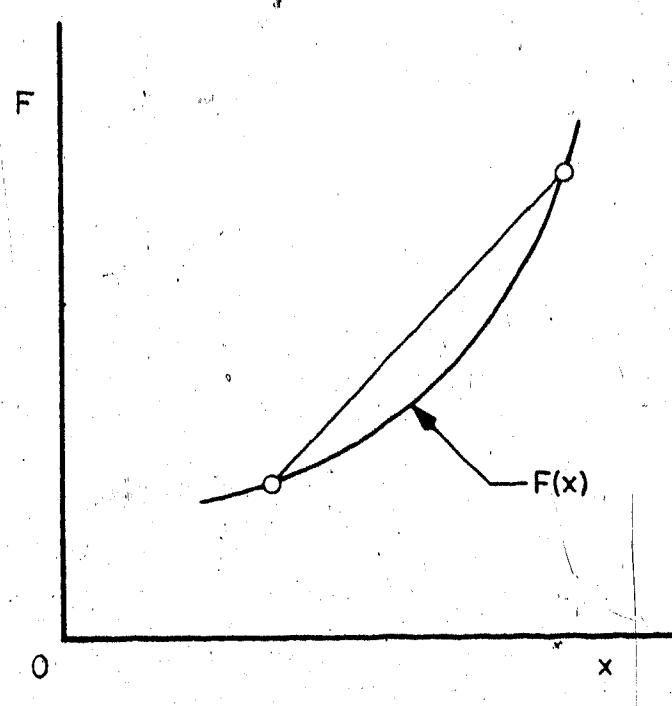
The approximation of (1.1) by (1.2) will only be effective over a small interval of displacement or time. The initial and final displacements of this interval will determine the length of the straight line approximation of the restoring force-displacement curve just mentioned. The slope of this line will be k in (1.2). If the damping expression in (1.1) is nonlinear, c would have to be found by iteration over the interval of displacement or time, depending upon the nature of the nonlinearity. The solution to (1.2) can now be obtained, using the initial conditions for the interval in question. The end conditions of this interval become the initial values of the one following, and the procedure is repeated with new values of c and k , as well as n and p , being determined.

By successfully solving (1.2) for each interval, an approximate

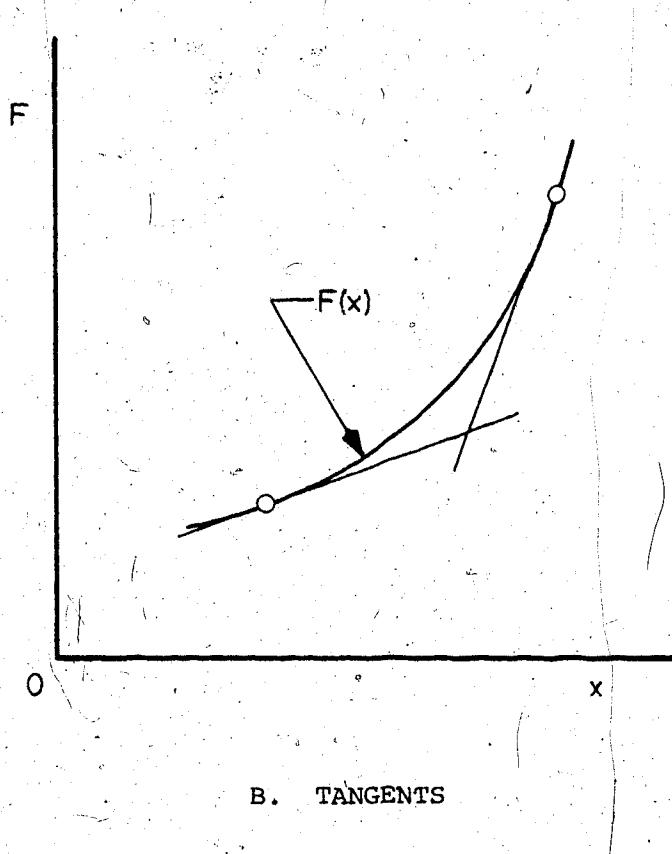
solution for (1.1) can be obtained, approaching the actual one as the intervals used become smaller.

The straight line approximation of the restoring force-displacement curve is accomplished by running a chord between the two endpoints of the section (Figure 1A) or tangents at the respective locations (Figure 1B). These points are located on the curve by knowing the initial and final displacements of the interval over which (1.2) is to be used. This force is shown in the figure as F and the displacement as x .

With this in mind, one can now apply this method to specific problems, as shall be seen in the following chapters.



A. CHORDS



B. TANGENTS

FIGURE 1 - METHODS OF PIECEWISE LINEARIZATION

II. HARD SPRING EQUATION (UNDAMPED)

A. Preliminary Comments

One of the simplest examples of the application of piecewise linearization is that of the hard undamped spring, the equation of which is:

$$x + ax + bx^3 = 0, \quad (2.1.1)$$

where:

$$a, b > 0. \quad (2.1.2)$$

See Figure 2.1 for details [7]. This is considered to be the equation for a hard spring since the restoring force increases quicker with respect to deflection than if it were linear [8]. The overall time span to be considered is the first quarter-cycle, since (2.1.1) is an odd function.

The following initial conditions will be used:

$$x(0) = x_0, \quad \dot{x}(0) = 0. \quad (2.1.3)$$

The cases examined had values for a and b of 1.0 and 0.15 respectively, and 1.0 and 2.0, respectively, to see how the solutions behaved with a low and a high degree of nonlinearity.

B. Approximation to Exact Solution

An exact solution for this equation exists, making use of Jacobian elliptic functions. The derivations of this result can be found in any

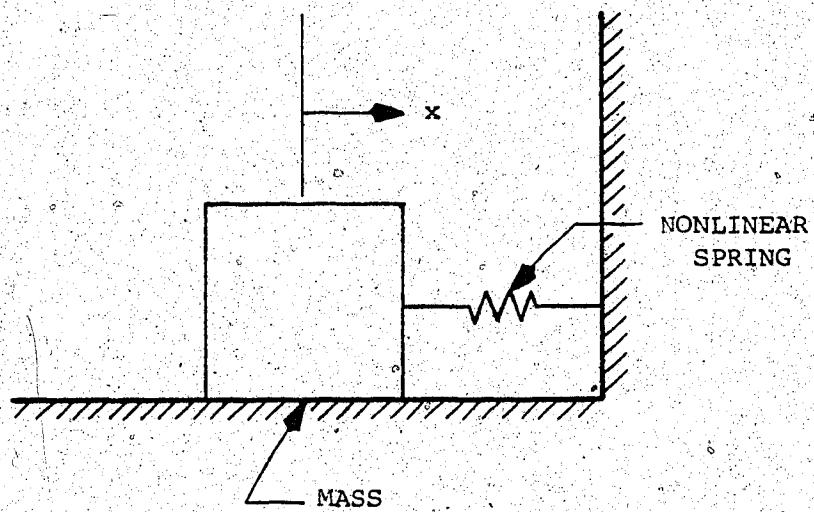


FIGURE 2.1 - MASS-SPRING SYSTEM (UNDAMPED)[7]

one of McLachlan [9], Cunningham [10], or Soudack [11]. This exact solution starts with:

$$\begin{aligned} t &= \int_0^\phi \frac{d\psi}{(1 - \lambda^2 \sin^2 \psi)^{1/2}} \\ &= \frac{F(\lambda, \phi)}{(a + b x_0^2)^{1/2}}, \end{aligned} \quad (2.2.1)$$

where:

$$\phi = \cos^{-1}\left(\frac{x}{x_0}\right),$$

$$\lambda^2 = \frac{b x_0^2}{2(a + b x_0^2)},$$

$F(\lambda, \phi)$ = Incomplete elliptical integral of the first kind.

By applying the assumptions (2.1.2) and initial conditions (2.1.3), the solution to (2.1.1) is, together with (2.2.1)[12], [13]:

$$x = x_0 \operatorname{cn}(\lambda \omega t), \quad (2.2.2)$$

where:

$$\omega^2 = (a + b x_0^2). \quad (2.2.3)$$

At the end of the first quarter-cycle, ϕ is $\pi/2$. Using the definition for some of the terms of (2.1.4), $F(\lambda, \phi)$ is known as the

complete elliptic integral of the first kind, which is given the designation of $K(\lambda^2)$ [14]. But, it is not always practical to refer to tables each time one varies a and/or b , and so an easier means of obtaining this value should be used. An infinite series does exist for this and for a specific value for λ^2 , one can use the following [15]:

$$\begin{aligned} K(\lambda^2) &= \frac{\pi}{2} \left[1 + \left(\frac{1}{2} \right)^2 (\lambda^2) + \left(\frac{1 \cdot 3}{2 \cdot 4} \right)^2 (\lambda^2) \right. \\ &\quad \left. + \left(\frac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6} \right)^2 (\lambda^2)^3 + \dots \right], \\ &(|\lambda^2| < 1), \end{aligned} \tag{2.2.4}$$

substituting this for $F(\lambda, \phi)$ in (2.2.1)[9], [10], [11] for calculating the period; and combining this with the definition for λ^2 to generate the solution. Since only the first 100 terms of (2.2.4) will be calculated, the solution obtained, which will be used as a benchmark for the piecewise linear results, will only be an approximation to (2.2.2) [12], [13] and shall be subsequently be referred to as such.

A method to generate the values for $cn(\lambda, wt)$ is required. One does exist, making use of what is known as the "arithmetic-geometric mean" approach [16]. The computer program that was used for solving (2.1.1) accessed a subroutine library devised by the computing centre at the University of British Columbia which uses this method [4].

From McLachlan [17], the period of oscillation is:

$$\tau_{\pi} = \frac{4K(\lambda^2)}{(a + bx_0^2)^{1/2}} \tag{2.2.5}$$

and so, for the quarter-cycle,

$$\tau_{\pi/4} = \frac{K(\lambda^2)}{(a + bX_0^2)^{1/2}} .$$

It should be noted that it is necessary to know the solution period as a basis for comparing the two piecewise linear methods that are to follow.

The actual solutions obtained by these methods are to be compared as well as with the one obtained by this approximation to the exact solution. The curve for the approximation to (2.2.2)[4], [15], [17] is obtained by dividing the quarter-cycle into n increments of time (where n is an integer) and calculating the displacement by incrementing the value of t by Δt , which is defined as follows:

$$\Delta t = \frac{K(\lambda^2)}{n(a + bX_0^2)^{1/2}} .$$

C. Derivation of Piecewise Linear Solution by Chords

The following method is based on Timoshenko, et. al. [18].

Assume that a restoring force-displacement curve for the system shown in Figure 2.1 can be approximated by a series of linear segments, as shown in Figure 2.2. The force is designated as F , while displacement is x . Assume further that the initial total displacement is divided into m equal sections of length Δx (m being an integer), with the total displacement being X_0 , or:

$$X_0$$

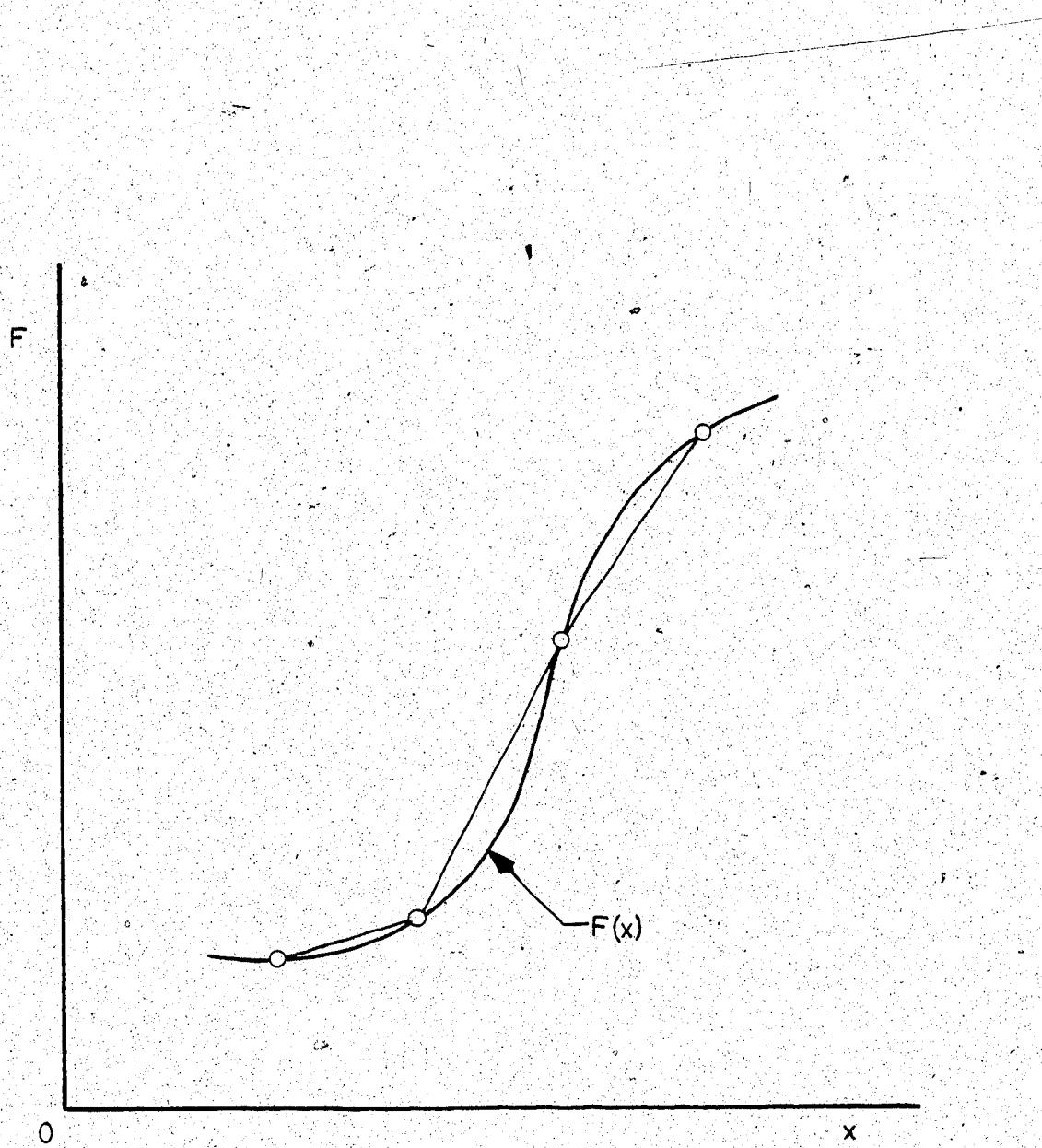


FIGURE 2.2 - APPROXIMATION BY CHORDS

Now consider, specifically for (2.1.1), two adjacent points (A and B) with displacements x_0 and x_1 , respectively, and times t_0 and t_1 , respectively (Figure 2.3). The system is initially at rest with displacement X_0 , and then released. For this:

x_0 = Initial displacement of segment

$$= x(t_0),$$

x_1 = Final displacement of segment

$$= x(t_1),$$

k_0 = Slope of segment,

F_0 = Restoring force at beginning of segment

$$= M(ax_0 + bx_0^3),$$

F_1 = Restoring force at end of segment

$$= M(ax_1 + bx_1^3),$$

x_0^* = x - axis crossing point for chord,

M = System mass.

It can be also seen that:

$$\Delta x = x_0 - x_1$$

and:

$$\begin{aligned} \frac{k_0}{M} &= \frac{F_0 - F_1}{M\Delta x} \\ &= \frac{a(x_0 - x_1) + b(x_0^3 - x_1^3)}{x_0 - x_1}. \end{aligned}$$

The equation of motion for line AB is (for $x_1 \leq x \leq x_0$, $t_1 \geq t \geq t_0$) [19]:

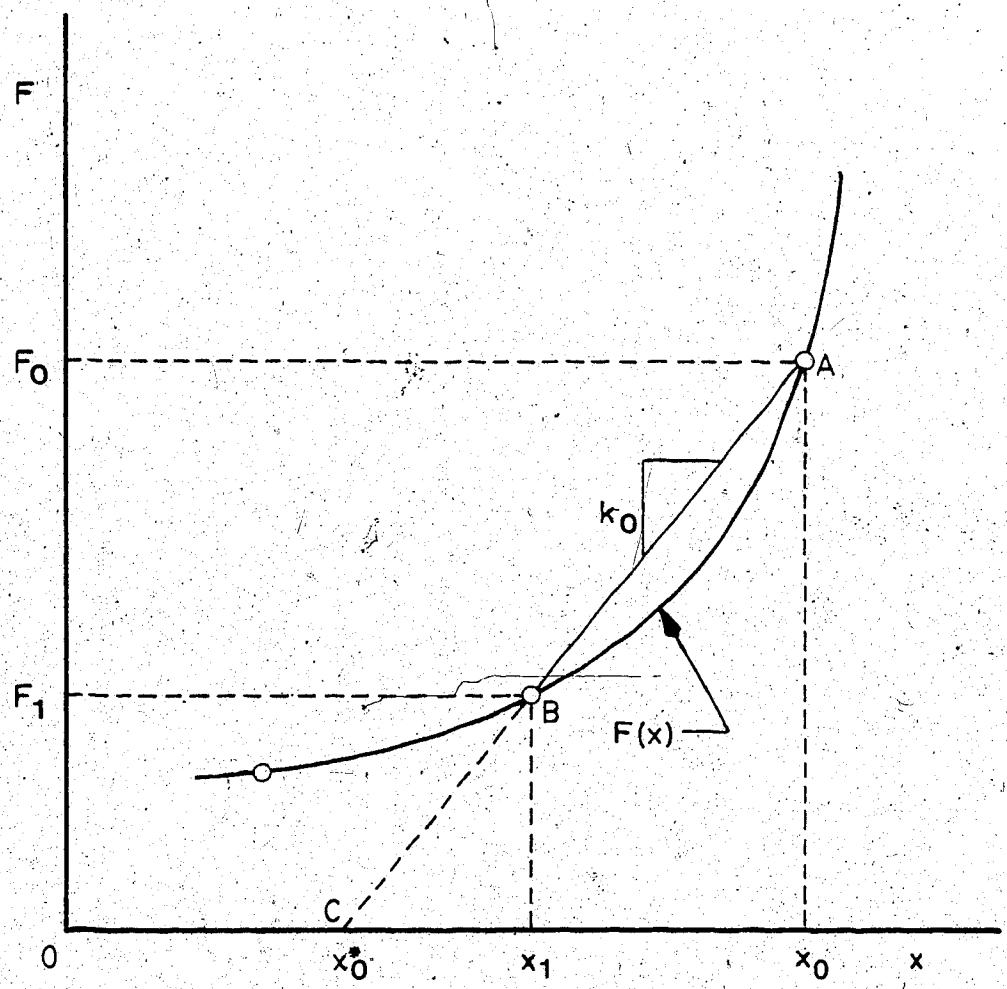


FIGURE 2.3 - CHORD SEGMENT CONSTRUCTION

$$\frac{d^2}{dt^2} (x - x_0^*) + \frac{k_0}{M} (x - x_0^*) = 0. \quad (2.3.1)$$

Referring to Figure 2.3, the value of x_0^* is determined as follows. AB is extended to the displacement axis (line AC) and the ratio of BC to AC is found. This is done by using the point-of-division methods from plane analytic geometry [20]:

$$\begin{aligned} r &= \frac{BC}{AC} \\ &= \frac{F_1 - 0}{F_0 - 0} \\ &= \frac{x_1 - x_0^*}{x_0 - x_0^*}, \\ (x_0 - x_0^*)r &= x_1 - x_0^*, \\ x_0^*(1 - r) &= x_1 - x_0^*r, \\ x_0^* &= \frac{x_1 - x_0^*r}{1 - r} \\ &= \left(\frac{1}{1 - F_1/F_0} \right) [x_1 - (F_1/F_0)x_0]. \end{aligned}$$

The solution to (2.2.1) is of the form [21]:

$$x = x_0^* + A \cos p(t - t_0) + B \sin p(t - t_0),$$

where:

$$p = \sqrt{\frac{k_0}{M}}.$$

Differentiating gives the velocity [21]:

$$\dot{x} = -pA \sin p(t - t_0) + pB \cos p(t - t_0).$$

Consider what happens at the beginning of the segment. The results are:

$$x_0 - x^* = A \cos p(0) + B \sin p(0),$$

$$A = x_0 - x^*,$$

$$\dot{x}_0 = -pA \sin p(0) + pB \cos p(0),$$

$$B = \frac{\dot{x}_0}{p}.$$

Therefore, the general solution for any segment would be [21]:

$$\begin{aligned} x &= x^* + (x_0 - x^*) \cos p(t - t_0) \\ &\quad + \frac{\dot{x}_0}{p} \sin p(t - t_0). \end{aligned} \tag{2.3.2}$$

with the velocity [21]:

$$\begin{aligned} \dot{x} &= -p(x_0 - x^*) \sin p(t - t_0) \\ &\quad + \frac{\dot{x}_0}{p} \cos p(t - t_0). \end{aligned} \tag{2.3.3}$$

For any given x_0 , the value of t_0 is the total time taken to traverse the preceding individual segments. The question now is what, for any given segment, this time would be. From (2.3.2), it can be seen that for a given segment, the time taken to go from x_0 to x_1 would be $(t_1 - t_0)$, where, in this case, t_1 is the total time taken to reach x_1 .

Making the substitution of t_1 for t in (2.3.2) yields the following:

$$\begin{aligned} x_1 &= x_0^* + (x_0 - x_0^*) \cos p(t_1 - t_0) \\ &\quad + \frac{\dot{x}_0}{p} \sin p(t_1 - t_0). \end{aligned}$$

This takes the form of the trigonometric equation [22]:

$$\gamma = \alpha \cos \theta - \beta \sin \theta,$$

where:

$$\left. \begin{aligned} \alpha &= x_0 - x_0^* \\ &> 0, \\ \beta &= \left| \frac{\dot{x}_0}{p} \right| \\ &> 0, \\ \gamma &= x_1 - x_0^*. \end{aligned} \right\} \quad (2.3.4)$$

Two possible solutions are possible for this equation [22]:

$$\theta_1 = -\tan^{-1}\left(\frac{\beta}{\alpha}\right) - \cos^{-1}\left(\frac{\gamma}{\sqrt{\alpha^2 + \beta^2}}\right),$$

$$\theta_2 = -\tan^{-1}\left(\frac{\beta}{\alpha}\right) + \cos^{-1}\left(\frac{\gamma}{\sqrt{\alpha^2 + \beta^2}}\right).$$

From $\tan^{-1} \delta = -\tan^{-1}(-\delta)$ [23] and the fact that $\gamma > 0$, plus $\theta > 0$ at a point right after $t = 0$,

$$\theta = p(t_1 - t_0)$$

$$= \tan^{-1} \left(-\frac{\beta}{\alpha} \right) + \cos^{-1} \left(\frac{\gamma}{\sqrt{\alpha^2 + \beta^2}} \right), \quad (2.3.5)$$

$$t_1 = t_0 + \frac{\theta}{p}.$$

using the definitions of (2.3.4).

Since the various values would have to be found iteratively, the quarter-period for this method would be the final value of t_1 .

D. Derivation of Piecewise Linear Solution by Tangents

As for the method of chords, the reference for this section is Timoshenko, et. al. [18].

Referring to the oscillator shown in Figure 2.1, consider now the case of the restoring force-displacement curve being approximated by a series of linear segments, but using tangents instead of chords (Figure 2.4). The force is designated as F and displacement as x . Assume further that, as before, the length of the segment is Δx . For two consecutive segments, specifically for (2.1.1), the construction would look like Figure 2.5, with:

x_0 = Displacement at beginning of segment,

x_1 = Displacement at end of segment,

x_0^* = x -axis crossing point for tangent,

x_{C0} = Point where tangents from x_0 and x_1 cross

= $x(t_{C0})$,

x_{C1} = Point where tangent from x_1 crosses one
from endpoint of next segment,

= $x(t_{C1})$,

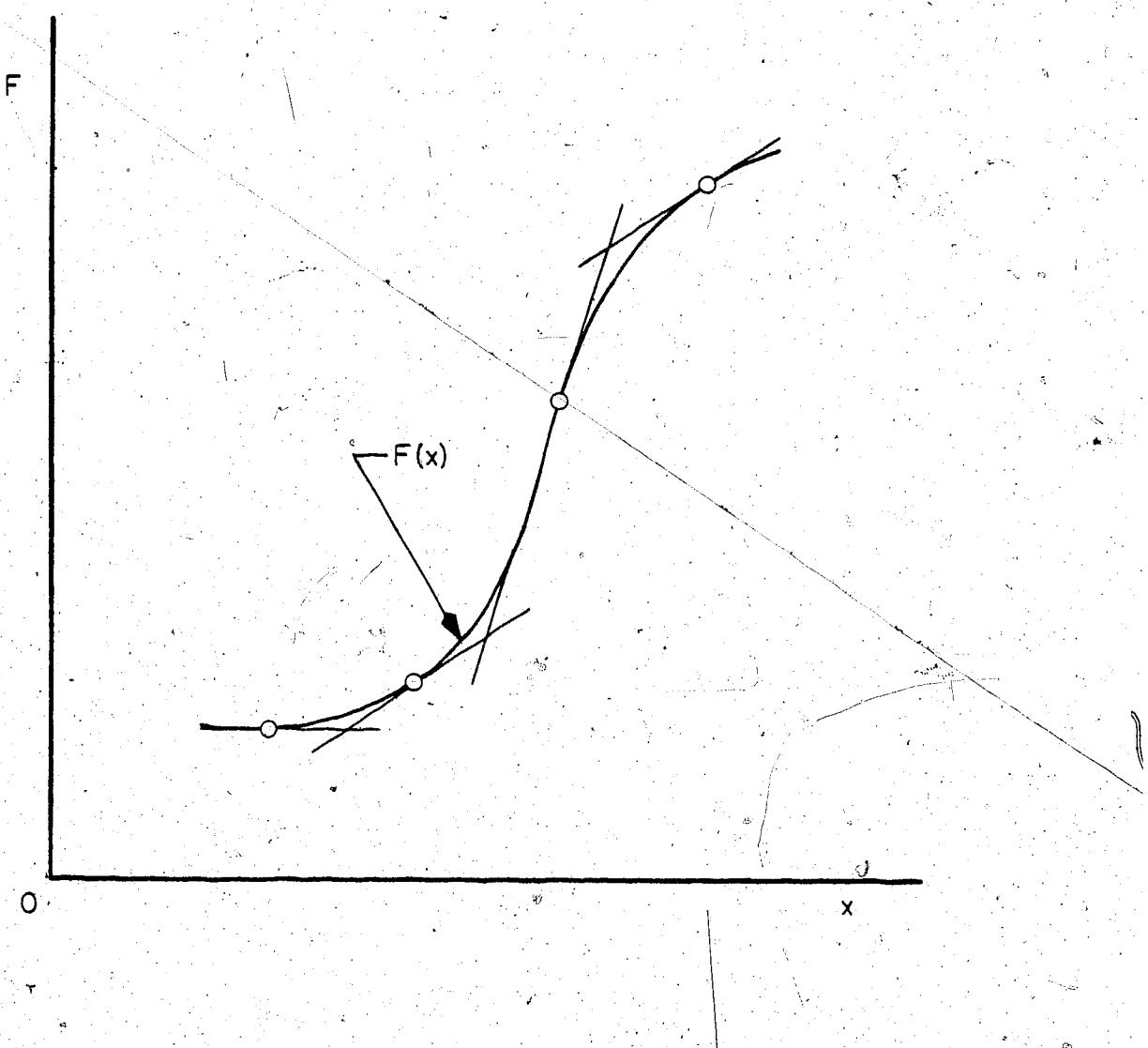


FIGURE 2.4 - APPROXIMATION BY TANGENTS

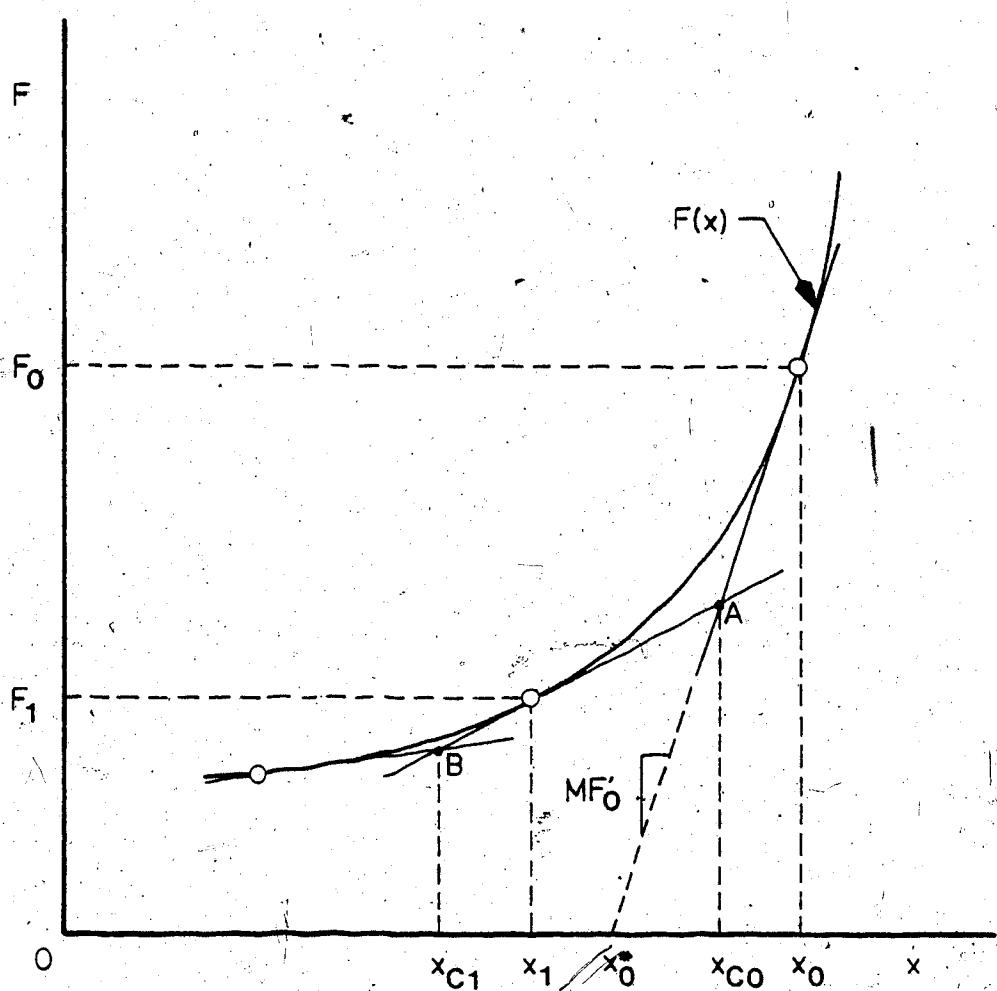


FIGURE 2.5 - TANGENT SEGMENT CONSTRUCTION

MF'_0 = Slope of tangent line at x_0 ,

F_0, F_1 = Same as for method of chords.

Again, as before for m equal sectors of length Δx (with m being an integer):

$$\Delta x = \frac{x_0}{m}$$

$$= x_0 - x_1.$$

In this case, motion is from one crossing point to the next (that is, from A to B), and so, the equation for line AB is (for $x_{C1} \leq x \leq x_{C0}$ and $t_{C1} \geq t \geq t_{C0}$) [19]:

$$\frac{d^2}{dt^2}(x - x_0^*) + \frac{k_0}{M}(x - x_0^*) = 0, \quad (2.4.1)$$

$$\frac{k_0}{M} = a + 3bx_0^2$$

$$= F'_0.$$

From Figure 2.5 it can be seen that:

$$F'_0 = \frac{F_0 - 0}{M(x_0 - x_0^*)}$$

$$= \frac{ax_0 + bx_0^3}{x_0 - x_0^*}$$

$$x_0^* = x_0 - \frac{ax_0 + bx_0^3}{a + 3bx_0^2}$$

The solution to (2.4.1) is on the form:

$$x = x_0^* + A \cos p(t - t_{C0}) + B \sin p(t - t_{C0}),$$

where:

$$p = \sqrt{\frac{F}{m}}.$$

The velocity is then [21]:

$$\dot{x} = -pA \sin p(t - t_{C0}) + pB \cos p(t - t_{C0}).$$

Using the same method as before, but for $t = t_{C0}$, the solution for line AB is [21]:

$$\begin{aligned} x &= x_0^* + (x_{C0} - x_0^*) \cos p(t - t_{C0}) \\ &\quad + \frac{\dot{x}_{C0}}{p} \sin p(t - t_{C0}), \end{aligned} \tag{2.4.2}$$

and the associated velocity [21]:

$$\begin{aligned} \dot{x} &= -p(x_{C0} - x_0^*) \sin p(t - t_{C0}) \\ &\quad + \frac{\dot{x}_{C0}}{p} \cos p(t - t_{C0}). \end{aligned} \tag{2.4.3}$$

Consider now what happens when the system reaches x_1 . In other words:

$$\begin{aligned} x_{C1} &= x_0^* + (x_{C0} - x_0^*) \cos p(t_{C1} - t_{C0}) \\ &\quad + \frac{\dot{x}_{C0}}{p} \sin p(t_{C1} - t_{C0}), \end{aligned}$$

$$\dot{x}_{C1} = -p(x_{C0} - x_0^*) \sin p(t_{C1} - t_{C0}) \\ + \dot{x}_{C0}^* \cos p(t_{C1} - t_{C0}).$$

The first expression takes on the form of the trigonometric equation referred to in the previous section [22], with the coefficients in this case being:

$$\left. \begin{aligned} \alpha &= x_{C0} - x_0^* \\ &> 0, \\ \beta &= \left| \frac{\dot{x}_{C0}}{p} \right| \\ &> 0, \\ \gamma &= x_{C1} - x_0^*. \end{aligned} \right\} \quad (2.4.4)$$

Again, there are two possibilities for a solution with the same logic as before, such that, in this case, the result is [22]:

$$\theta = \tan^{-1} \left(-\frac{\beta}{\alpha} \right) + \cos^{-1} \left(\frac{\gamma}{\sqrt{\alpha^2 + \beta^2}} \right) \\ = p(t_{C1} - t_{C0}), \\ t_1 = t_0 + \frac{\theta}{p}. \quad (2.4.5)$$

As before, the various segments have their individual values for t_1 , and so, the quarter-period is the sum of all of these.

However, there is a slight difference here, owing to the fact that at each point, a tangent is drawn. The endpoints will not have full tangent lines, as can be seen by inspection. For the initial

displacement, the equation of motion will be:

$$x = x_0^* + (x_{C0}^* + x_0^*) \cos pt.$$

For the end of the quarter period (or $x_1 = 0$), the result is:

$$\begin{aligned} -x_0^* &= (x_{C0}^* - x_0^*) \cos p(t_{C1} - t_{C0}) \\ &\quad + \frac{\dot{x}_{C0}}{p} \sin p(t_{C1} - t_{C0}). \end{aligned}$$

Each x_{C0}^* is found as follows. From plane analytic geometry [24], for an x-y plot and two points (x_1, y_1) and (x_2, y_2) with the respective slopes being:

$$m_1 = \frac{y - y_1}{x - x_1},$$

$$m_2 = \frac{y - y_2}{x - x_2},$$

$$\begin{aligned} y &= m_1 x + (-m_1 x_1 + y_1), \\ &= m_2 x + (-m_2 x_2 + y_2). \end{aligned}$$

Extending this to the current situation:

$$F = F'_0 x_{C0} - F'_0 x_0 + F_0$$

$$= F'_1 x_{C0} - F'_1 x_1 + F_1,$$

$$(F'_0 - F'_1) x_{C0} = F'_0 x_0 - F'_1 - (F_0 - F_1),$$

$$x_{C0} = \frac{F'_0 x_0 - F'_1 x_1 - (F_0 - F_1)}{F'_0 - F'_1}.$$

where:

$$F_1' = a + 3x_1^2.$$

The two methods of solution had used different quantities for incrementation: time for the approximation to the exact solution and displacement for the piecewise linearization. For the former, time was the better of the two, as it is one variable of the argument. For the piecewise linearization, it was easier to use displacement, since one can readily find the values for x_0 , k_0 , and so on, since these are based upon a value for Δx . By using time, this does not become immediately apparent, and it would be difficult to determine the end of an interval. First the end of the segment would have to be guessed, and all the other variables used in the construction of a segment calculated on that basis. A variable, such as the current average segment velocity, is used as a means of determining the end of the interval by comparing it to the previous segment velocity, adjusting the end of the segment until there is an agreement, recalculating all the various variables concerned each time. This method is long and complicated and would increase the cost of the run. Also, the reference of Timoshenko, et. al. [18] already had a form of a solution that could readily be applied to this situation.

E. Results

A computer program was written using the three solutions, and the results will now be examined. This was run on the Amdahl 470V/8 at the University of Alberta [25], [26].

First consider the periods for each solution. It is important to consider this now as one is interested in the accuracy of the values obtained as compared with those from the approximation to the exact

TABLE 2.1 - UNDAMPED HARD SPRING EQUATION PERIODS
FOR $a = 1.0$, $b = 0.15$

Approx. to Exact Soln. = 5.958299670

<u>No. of Divisions</u>	<u>PIECEWISE LINEAR</u>	
	<u>CHORDS</u>	<u>TANGENTS</u>
5	5.951629401	5.961481246
10	5.956522358	5.959160535
15	5.957490299	5.958695097
20	5.957838384	5.958526637
25	5.958002069	5.958447100
30	5.958091944	5.958403305
35	5.958146564	5.958376633
40	5.958182232	5.958359186
45	5.958206807	5.958347149
50	5.958224456	5.958338494
55	5.958237560	5.958332062
60	5.958247556	5.958327151
65	5.958255356	5.958323316
70	5.958261559	5.958320265
75	5.958266573	5.958317797
80	5.958270684	5.958315772
85	5.958274097	5.958314090
90	5.958276961	5.958312678
95	5.958279389	5.958311481
100	5.958281464	5.958310457

(which uses the routine described in [4]). The first 100 terms of (2.2.4) were calculated for this. Initially, a low degree of nonlinearity was examined, with the results on Table 2.1. One can see, as was originally stated in the introduction, that as the number of segments gets large, the solution by chords and by tangents slowly approach the same value. Further evidence of this can be seen in Table 2.2, where the nonlinearity is more pronounced.

In theory, then, one could keep decreasing the size of the increments, and the periods would slowly converge to that of the exact solution to as many significant figures as is needed. The only limitation to this accuracy would be the execution time for the program, as each set of calculations will add to the run cost.

It should be noted that the period for the approximation to the exact solution was obtained as explained earlier in this chapter, while for the piecewise methods, the final value of t_1 was multiplied by 4.

An evaluation of the methods will be helped by examining the solution curves. The program that produced the results for Tables 2.1 and 2.2 also drew a plot each time it looped through, using the number of increments that was in effect as a basis.

The plots can be seen in Figures 2.6 through 2.9. A few comments should be made concerning the results. A good agreement to the approximation to the exact solution [4], [15], [17] (for $a = 1.0$ and $b = 0.15$) is obtained by both methods for 5 increments (Figure 2.6) and with 10 (Figure 2.7). Here one can hardly detect any difference between the piecewise method results and that of the approximation to (2.2.2) [4], [15], [17]. As one might suspect for $a = 1.0$ and $b = 2.0$, Figure 2.8 shows that there is a discernable deviation from the solution being

TABLE 2.2 - UNDAMPED HARD SPRING EQUATION PERIODS
FOR $a = 1.0$, $b = 2.0$

Approx. to Exact Soln. = 4.004308722

<u>No. of Divisions</u>	<u>PIECEWISE LINEAR</u>	
	<u>CHORDS</u>	<u>TANGENTS</u>
5	3.976017631	4.005261005
10	3.996812433	4.007978008
15	4.000907203	4.005978416
20	4.002378678	4.005261005
25	4.003061288	4.004923996
30	4.003438030	4.004739061
35	4.003666618	4.004626708
40	4.003815695	4.004553357
45	4.003918296	4.004502829
50	4.003991920	4.004466543
55	4.004046539	4.004439606
60	4.004088177	4.004419059
65	4.004120645	4.004403028
70	4.004146464	4.004390280
75	4.004167307	4.004379975
80	4.004184398	4.004371526
85	4.004198581	4.004364513
90	4.004210479	4.004358627
95	4.004220560	4.004353639
100	4.004229175	4.004349375

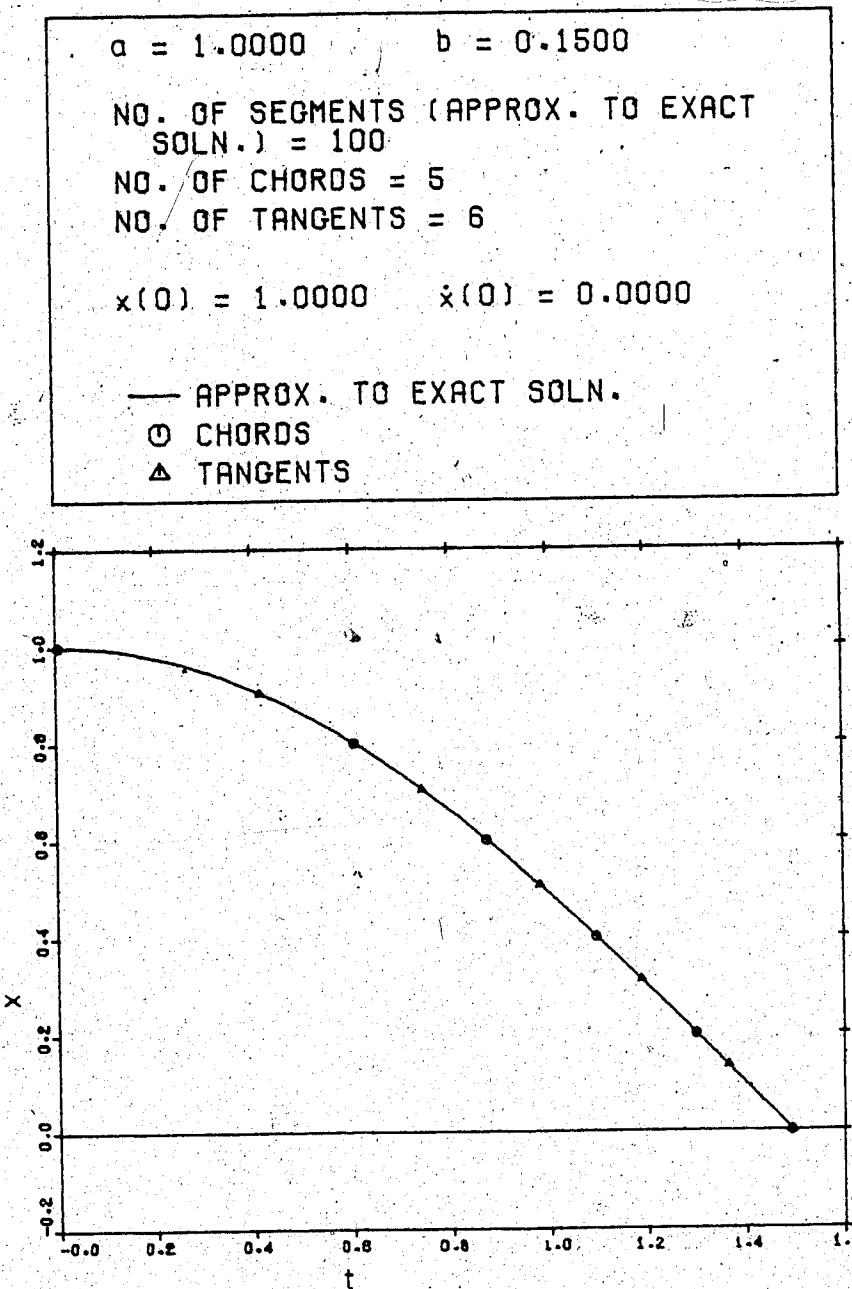


FIGURE 2.6 - SOLUTIONS TO HARD SPRING EQUATION
 (UNDAMPED) FOR $a = 1.0$, $b = 0.15$, $\Delta t = \tau_{\pi}/4/100.0$,
 $\Delta x = 0.20$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Approximation to exact solution uses [4], [15], [17].)

$a = 1.0000$ $b = 0.1500$
 NO. OF SEGMENTS (APPROX. TO EXACT
 SOLN.) = 100
 NO. OF CHORDS = 10
 NO. OF TANGENTS = 11
 $x(0) = 1.0000$ $\dot{x}(0) = 0.0000$
 — APPROX. TO EXACT SOLN.
 ○ CHORDS
 △ TANGENTS

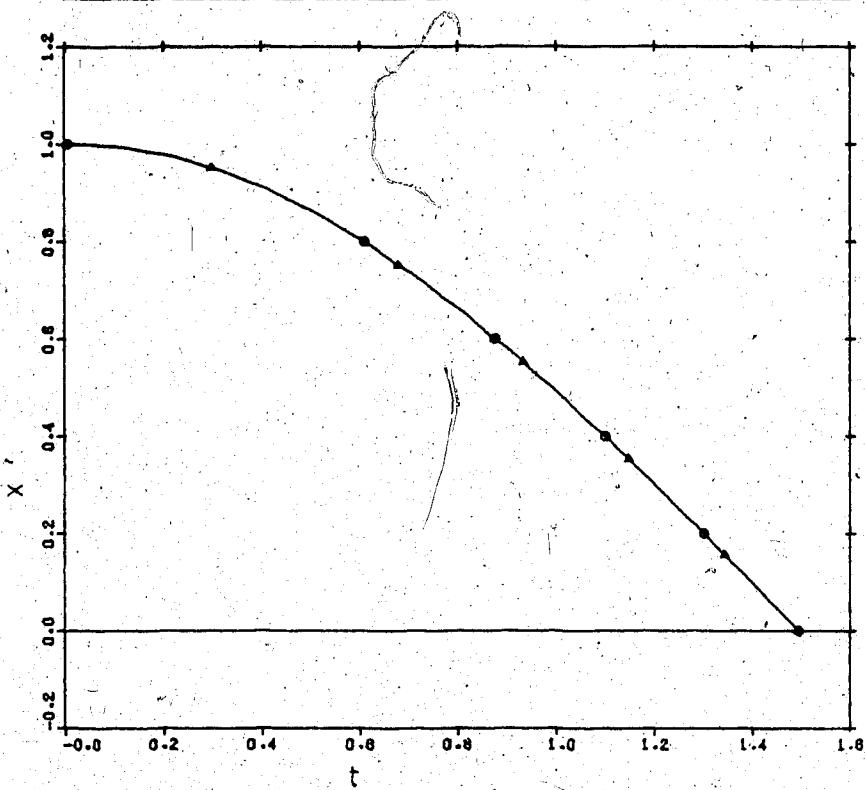


FIGURE 2.7 - SOLUTIONS TO HARD SPRING EQUATION
 (UNDAMPED) FOR $a = 1.0$, $b = 0.15$, $\Delta t = \pi/4 / 100.0$,
 $\Delta x = 0.10$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Approximation to exact solution uses [4], [15], [17].)

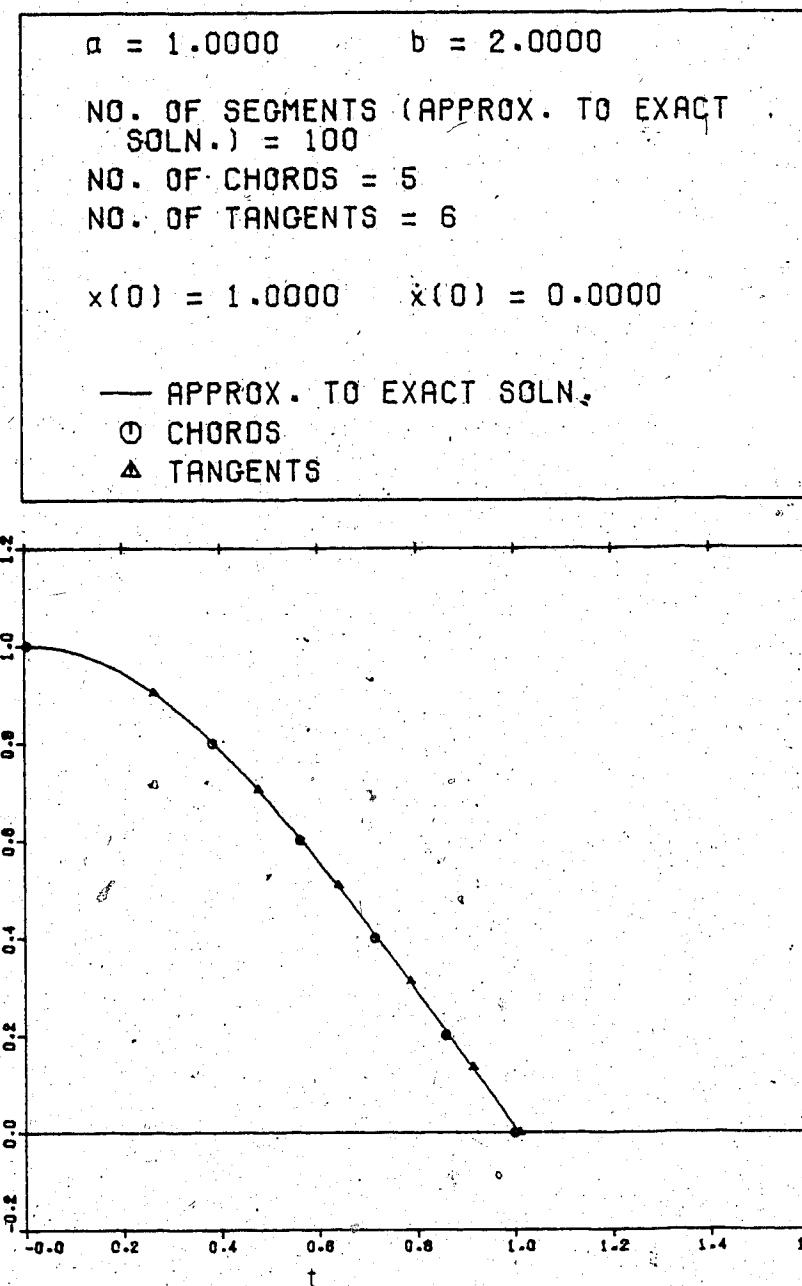


FIGURE 2.8 - SOLUTIONS TO HARD SPRING EQUATION
 (UNDAMPED) FOR $a = 1.0$, $b = 2.0$, $\Delta t = \tau_{\pi/4} / 100.0$,
 $\Delta x = 0.20$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Approximation to exact solution uses [4], [15], [17].)

$a = 1.0000$ $b = 2.0000$
 NO. OF SEGMENTS (APPROX. TO EXACT
 SOLN.) = 100
 NO. OF CHORDS = 10
 NO. OF TANGENTS = 11
 $x(0) = 1.0000$ $\dot{x}(0) = 0.0000$
 — APPROX. TO EXACT SOLN.
 ○ CHORDS
 △ TANGENTS

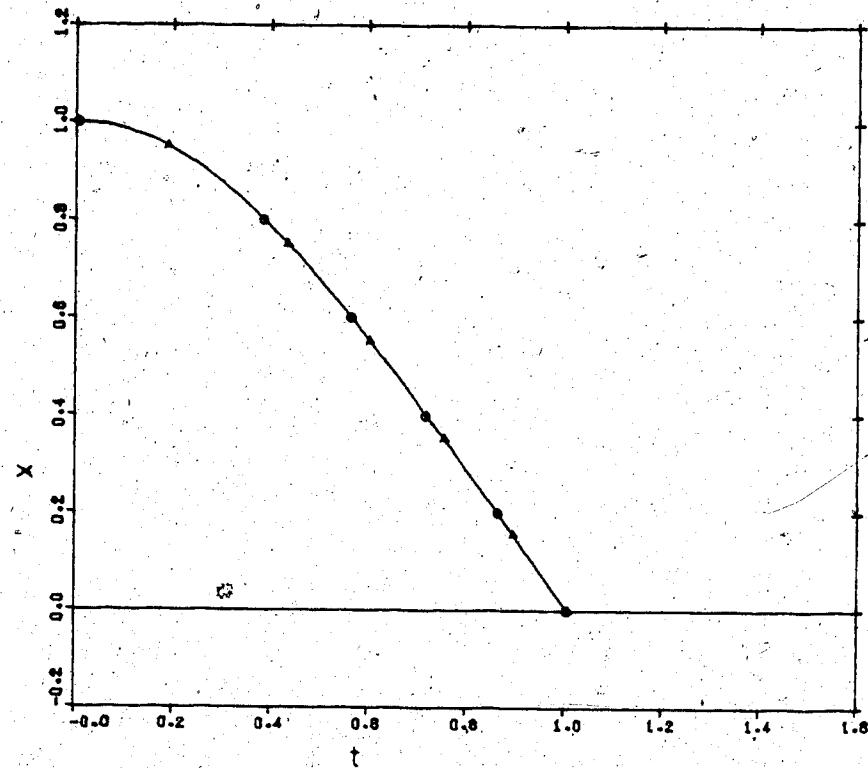


FIGURE 2.9 - SOLUTIONS TO HARD SPRING EQUATION
 (UNDAMPED) FOR $a = 1.0$, $b = 2.0$, $\Delta t = \pi/4$, 100.0 ;
 $\Delta x = 0.10$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Approximation to exact solution uses [4], [15], [17].)

compared, since the degree of nonlinearity is considerably higher than in the first case. This does not appear in Figure 2.9, where 10 increments were used.

A question that should be raised is concerning the actual execution times involved, since, when choosing a method of solving these sorts of problems, one must strike a balance between required accuracy to the solution being compared and the cost of the run. This was done by taking the original program and modifying it by stripping it of all non-essential operations with only the actual calculations remaining, and then utilizing a timing subroutine available through the library of the computer system, and calling it for each solution [27]. The results are seen on Figures 2.10 and 2.11. It should be noted here that these results were obtained with an older and somewhat slower version of the program, but they should serve to illustrate the relative execution times. It should also be noted that these results were obtained with an Amdahl 470V/7 [25]. This difference in the computers was due to an upgrading of the original 470V/7 model to a 470V/8 done while the research for this thesis was being carried out [26]. The author reran the timing program and found that the results were about 10 percent faster with the V/8 than the V/7. Also, with some further minor modifications to the program itself, the author estimates about another 10 percent can be taken off the original times.

By inspection, one can see that the actual distance travelled by the system using the method of chords would be somewhat less than that using tangents. Already this implies that the period would be greater using the latter method than the former when comparing it to that obtained by using the results based on the exact solution. That would

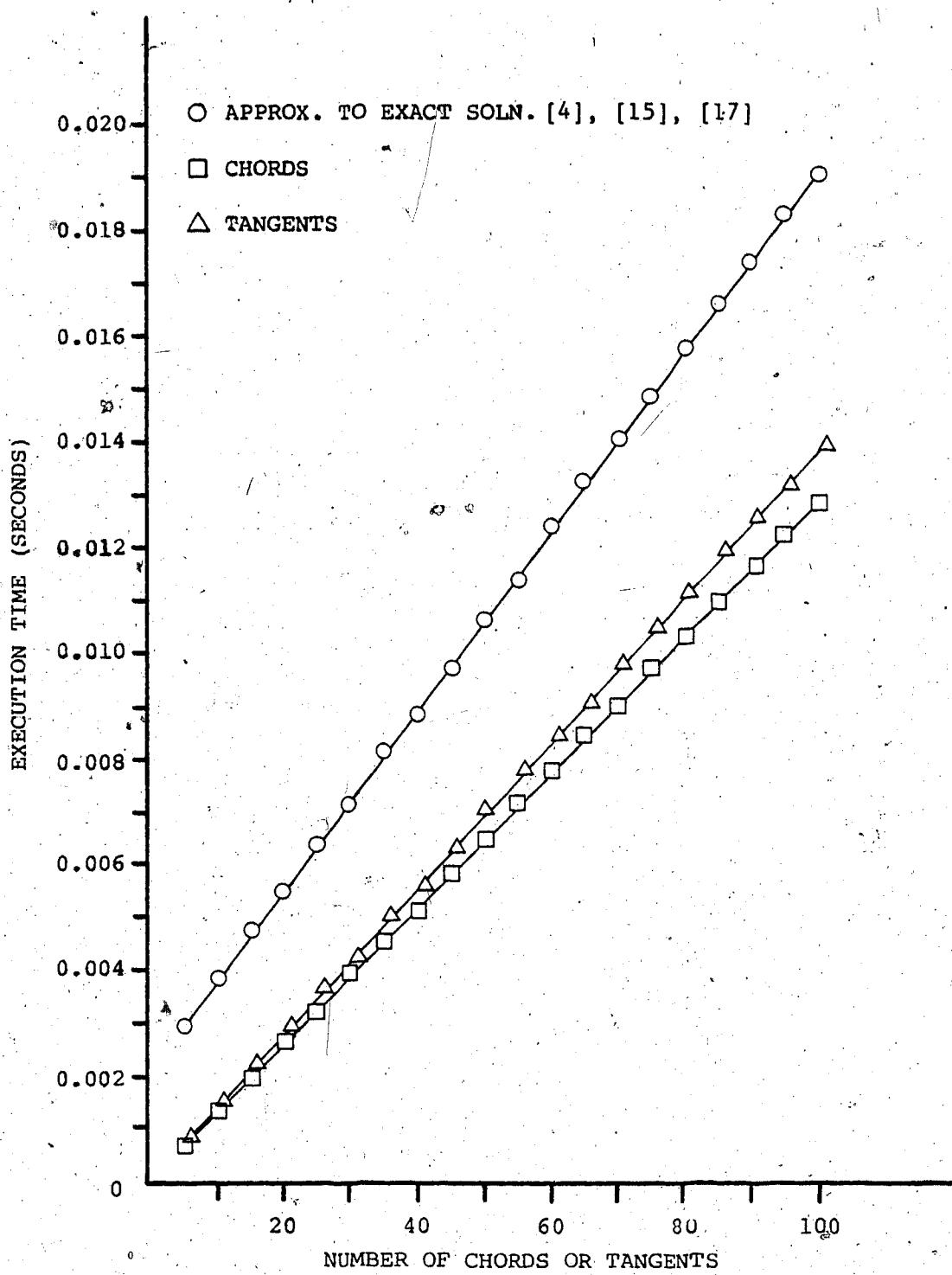


FIGURE 2.10 - EXECUTION TIMES FOR SOLUTIONS TO HARD SPRING EQUATION (UNDAMPED) FOR $a = 1.0$, $b = 0.15$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$ [27]

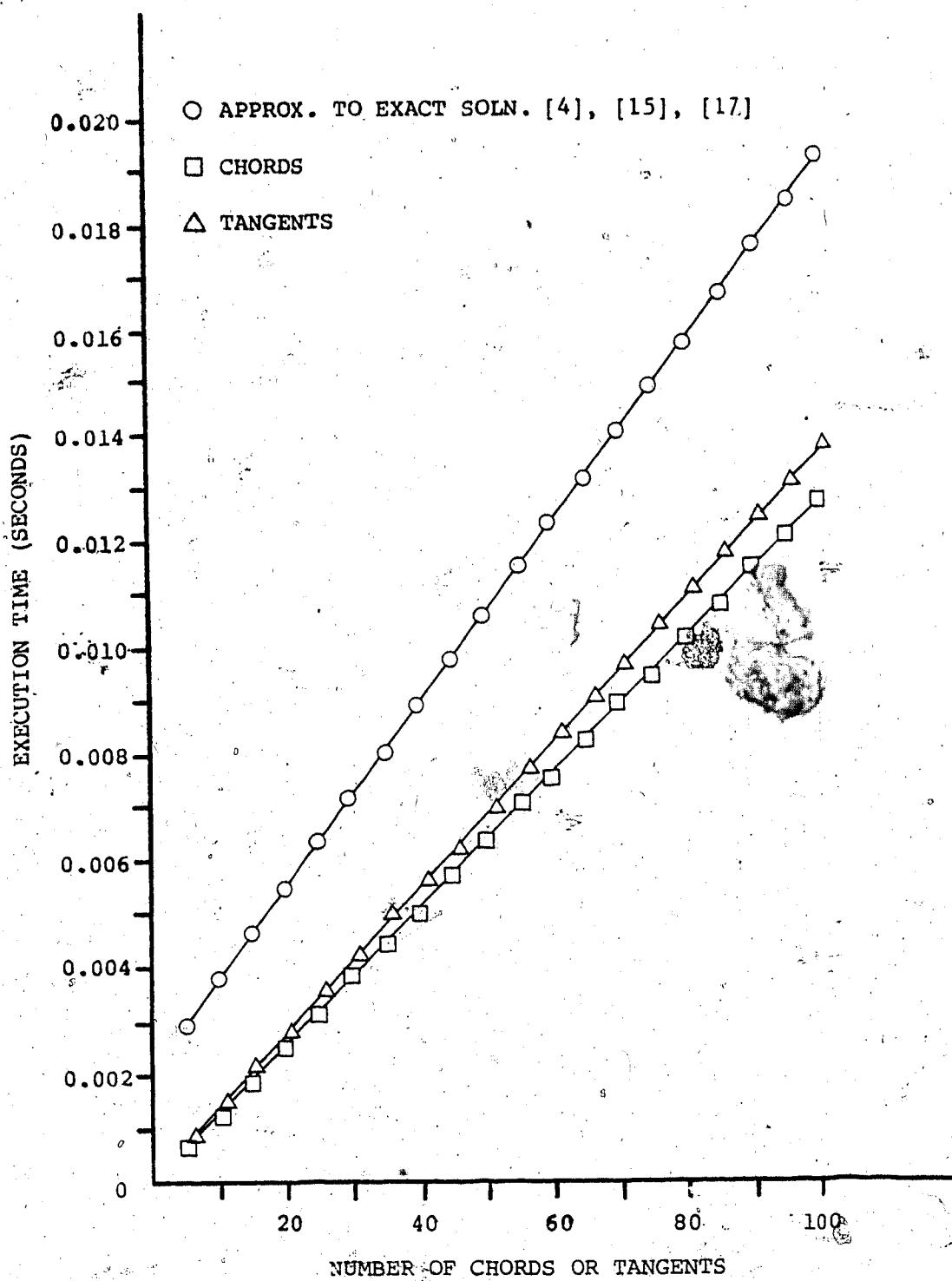


FIGURE 2.11 - EXECUTION TIMES FOR SOLUTIONS TO HARD SPRING EQUATION (UNDAMPED) FOR $a = 1.0$, $b = 2.0$,
 $x(0) = 1.0$, $\dot{x}(0) = 0.0$ [27]

explain why, when b was 2.0, for example, one obtained a difference in the displacement on the higher side using tangents than chords.

Another question is why the timing results for the approximation to the exact solution [4], [15], [17] should be considerably higher than that for the piecewise methods. A word of explanation concerning the program will clarify this point. The execution times were obtained by calculating the first 50 terms of (2.2.4)[15] and then generating the solution results, prior to increasing the number of segments and looping through the calculations once again. This was done to determine the total time required to obtain a solution for a given Δx , thus allowing comparison with the other methods. This required that the subroutine that calculated (2.2.4)[15] had to be called each time.

Since more calculations are necessary for each segment for the method of tangents, it would be expected that the execution times would be greater for this solution than for the method of chords.

In any case, the results for the piecewise methods are impressive when one considers that for this problem, it would appear that an accurate solution can be easily obtained by using either method, with a relatively small number of segments, and that it is faster than the solution that was chosen to be the standard one [4], [15], [17]. One can conclude that for this problem, the piecewise method would be the recommended one to use, particularly the method by chords.

Because the method of chords was easier to manipulate and converged faster to the approximation to (2.2.2)[4], [15], [17], it was chosen to be the method of piecewise linearization for the remainder of the thesis and will subsequently be referred to as such.

III. HARD SPRING EQUATION (LINEAR DAMPING)

A. Preliminary Comments

Most physical systems involve dissipation, and an example of this is the system described in the previous chapter, but with the inclusion of a simple form of damping in which it is linearly related to the velocity:

$$\ddot{x} + \frac{c}{M} + ax + bx^3 = 0, \quad (3.1.1)$$

where:

c = Damping constant,

M = System mass,

a, b = Restoring force factors, as used in

Chapter II.

The system is shown in Figure 3.1 [28].

Since there is no exact solution and approximate methods such as those mentioned in the first chapter are beyond the scope of this thesis, a Runge-Kutta solution [5] has been chosen to be the standard by which to judge the accuracy of the piecewise method.

For convenience, it was decided to restrict the cases examined to underdamped conditions.

The values for a and b are the same as in the previous chapter, while $\frac{c}{M}$ will have the values of 0.0, 0.1, and 0.25 which should give a good representation of an underdamped case.

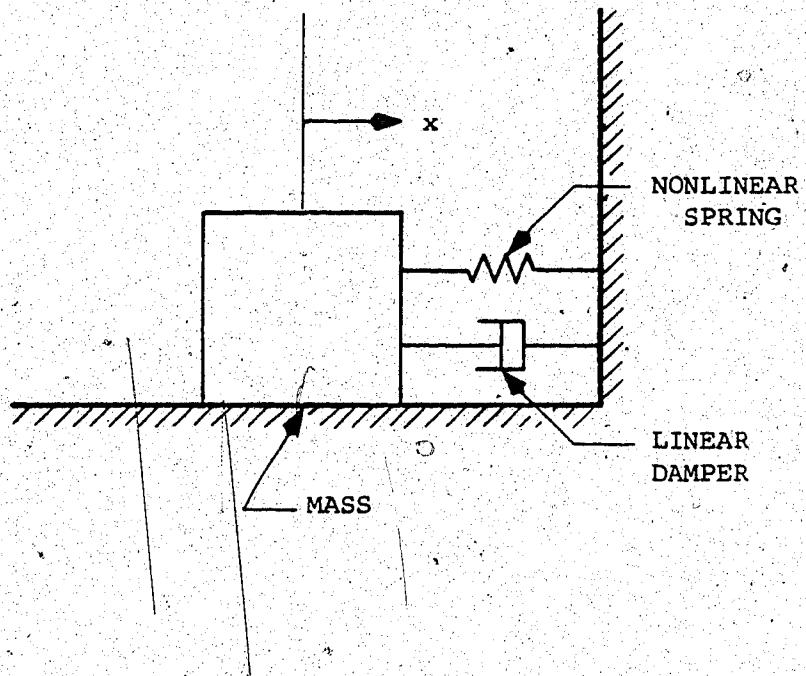


FIGURE 3.1 - MASS-SPRING SYSTEM (LINEAR DAMPING) [28]

B. Derivation of Piecewise Linear Solution

Since the piecewise linearization method by chords had been chosen to be the means of solving the equations for the remainder of the thesis, and much of the background derivation is exactly the same as before, only the results of this plus the derivation of new material will be shown in this and subsequent chapters.

Recalling the development of the method by chords and using the same initial conditions and notation as in the previous chapter, the following equation results for a linear segment [19], [29]:

$$\frac{d^2}{dt^2}(x - x_0^*) + \frac{C}{M}\dot{x} + \frac{k_0}{M}(x - x_0^*) = 0,$$

k_0 = Slope of chord line, as in Chapter II.

where x , x_0^* , and \dot{x} are the same as before and are found in the manner previously described in Chapter II, Section C. The solution to this system is now also familiar [29]:

$$x = x_0^* + e^{-n(t - t_0)} [A \cos p^*(t - t_0) + B \sin p^*(t - t_0)] \quad (3.2.1)$$

where:

$$p^* = \sqrt{\frac{k_0}{M} - n^2},$$

t_0 = Time taken for system to reach current x_0 ,

$$n = \frac{C}{2M},$$

$$B = \frac{\dot{x}_0 + n(x_0 - x_0^*)}{p^*}$$

Differentiating (3.2.1) yields the velocity [29]:

$$\begin{aligned} \dot{x} &= e^{-n(t - t_0)} \left[(p^*B - nA) \cos p^*(t - t_0) \right. \\ &\quad \left. + (-nB - p^*A) \sin p^*(t - t_0) \right], \end{aligned} \quad (3.2.2)$$

with the various coefficients as before.

Solving for $t_1 - t_0$ requires the use of a Newton-Raphson method, since there does not appear to be a simpler way of finding this quantity such as that used in the previous chapter. This would be [30]:

$$(t_1 - t_0)_{i+1} = (t_1 - t_0)_i - \frac{x[(t_1 - t_0)_i]}{\dot{x}[(t_1 - t_0)_i]}.$$

However, as the displacement approaches the first trough, the velocity approaches zero, and consequently the method as defined would break down. But, since this can be set up in a computer program, a check can be run such that when the denominator becomes very small (corresponding to a small speed) the Newton-Raphson relationship can be redefined as [30]:

$$(t_1 - t_0)_{i+1} = (t_1 - t_0)_i - \frac{\dot{x}[(t_1 - t_0)_i]}{x[(t_1 - t_0)_i]}.$$

$$x[(t_1 - t_0)_i] = e^{-n[(t_1 - t_0)_i]} \cdot \{ [(-p^*{}^2 + n^2)A - 2np^*B] \cos p^*[(t_1 - t_0)_i] \\ + [2np^*A + (-p^*{}^2 + n^2)B] \sin p^*[(t_1 - t_0)_i] \},$$

with the coefficients the same as used for (3.2.1) and (3.2.2).

Also, there is the likelihood that the first trough to be encountered bottoms out in the middle of a segment, which poses another problem as to how to determine exactly where this occurs. First $t_1 - t_0$ is found as before (with the second Newton-Raphson relationship being in effect as well as x will approach zero). The appropriate value for the displacement is found, and this is compared to what the particular x_1 would have been. If the difference between the two values is small, then the iterations stops. If not, this x becomes the new x_1 , and new end conditions are found with the iteration continuing using $t_1 - t_0$.

The half-period, then, would be the final value of t_1 .

C. Runge-Kutta Solution

The Runge-Kutta numerical method used throughout this thesis is the routine DVERK from IMSL [5]. Since it was in a system library, it was easily accessed by the program written to solve this situation. The basis of the method is an approach using approximations described in this reference.

D. Results

A computer program was written that included the piecewise linear solution, complete with the necessary subroutines for calculating the

FIGURE 3.2 - SOLUTIONS TO HARD SPRING EQUATION
(LINEAR DAMPING) FOR $a = 1.0$, $b = 0.15$, $\frac{c}{m} = 0.0$,
 $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5].)

$a = 1.0000 \quad b = 0.1500$
 $c/M = 0.0000$

Δx (PIECEWISE
LINEARIZATION) = 0.1000
 Δt (RUNGE-KUTTA) = 0.0500

$x(0) = 1.0000 \quad \dot{x}(0) = 0.0000$

— RUNGE-KUTTA
○ PIECEWISE LINEARIZATION

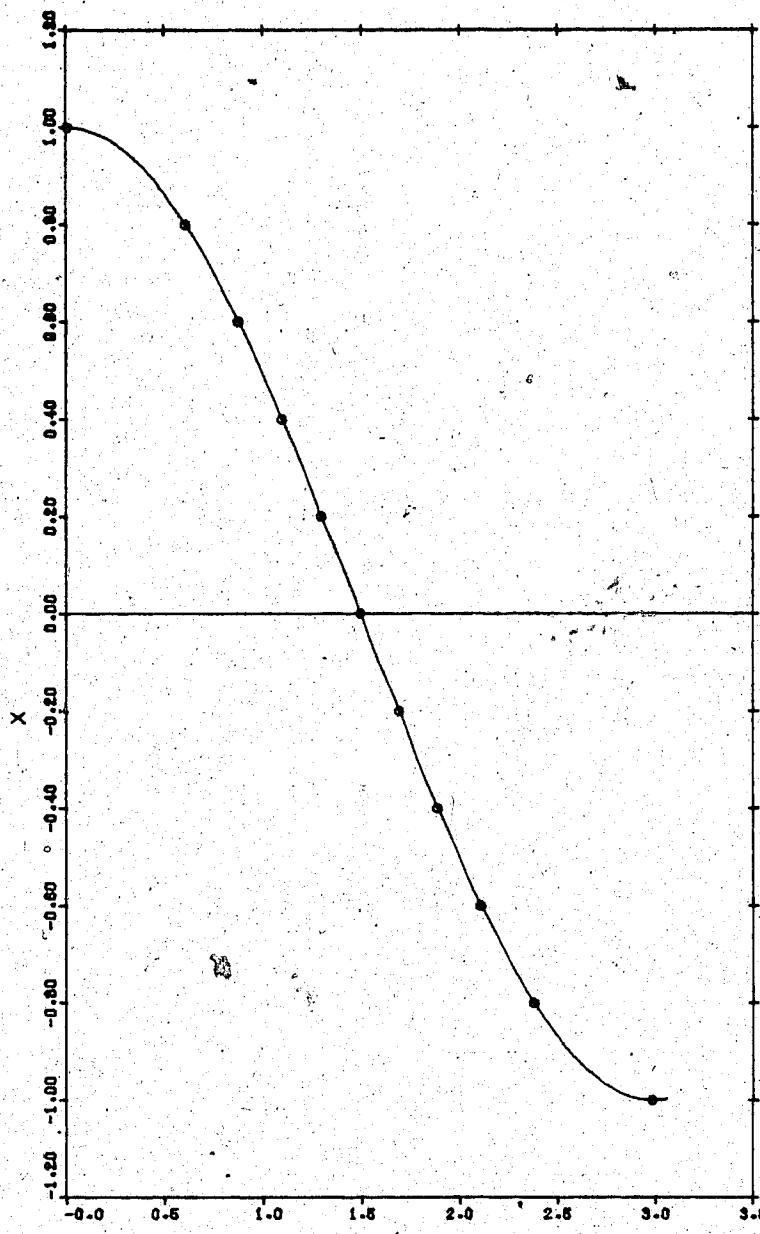


FIGURE 3.3 - SOLUTIONS TO HARD SPRING EQUATION
(LINEAR DAMPING) FOR $a = 1.0$, $b = 2.0$, $\frac{c}{M} = 0.0$,
 $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5].)

$a = 1.0000$ $b = 2.0000$
 $c/M = 0.0000$
 Δx (PIECEWISE
LINEARIZATION) = 0.1000
 Δt (RUNGE-KUTTA) = 0.0500
 $x(0) = 1.0000$ $\dot{x}(0) = 0.0000$

 — RUNGE-KUTTA
 ○ PIECEWISE LINEARIZATION

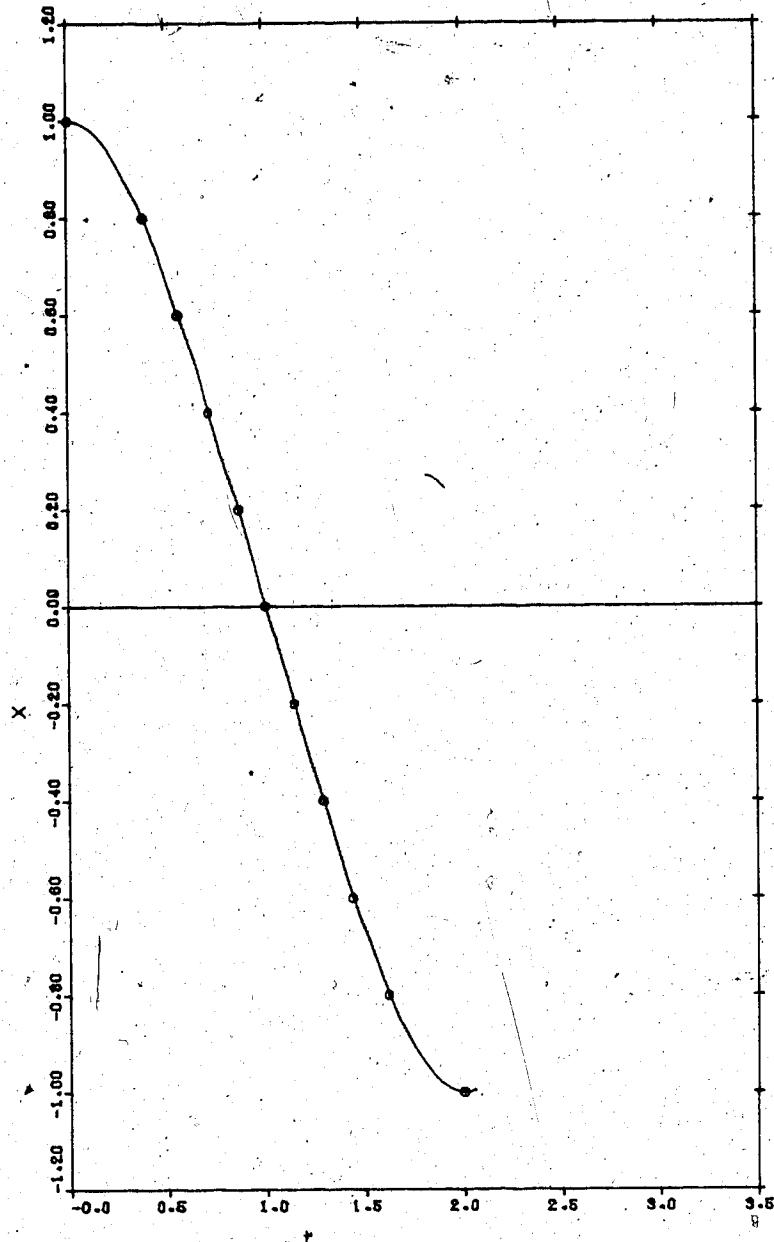


FIGURE 3.4 - SOLUTIONS TO HARD SPRING EQUATION
(LINEAR DAMPING) FOR $a = 1.0$, $b = 0.15$, $\frac{c}{m} = 0.10$,

$\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5].)

a = 1.0000 b = 0.1500
c/M = 0.1000

Δx (PIECEWISE LINEARIZATION) = 0.1000
 Δt (RUNGE-KUTTA) = 0.0500

$x(0) = 1.0000$ $\dot{x}(0) = 0.0000$

— RUNGE-KUTTA
○ PIECEWISE LINEARIZATION

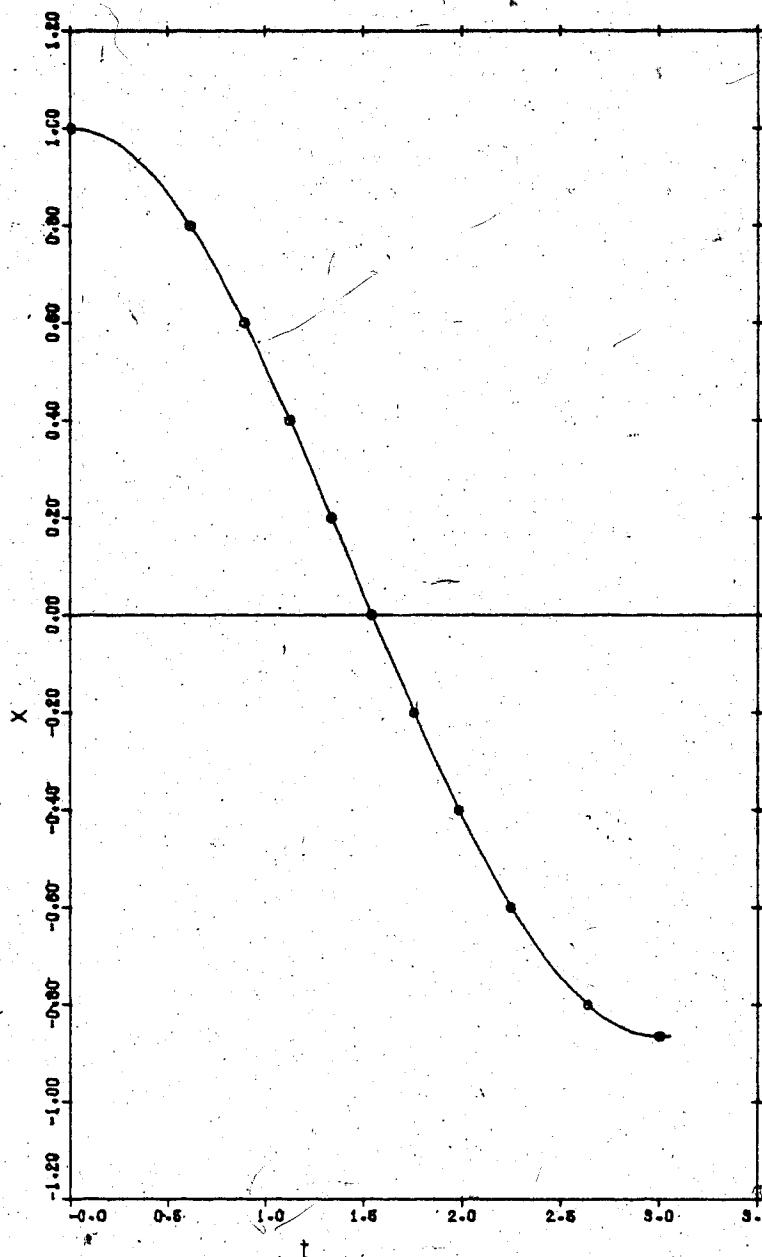


FIGURE 3.5 - SOLUTIONS TO HARD SPRING EQUATION
(LINEAR DAMPING) FOR $a = 1.0$, $b = 2.0$, $\frac{c}{m} = 0.10$,
 $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5].)

$a = 1.0000 \quad b = 2.0000$
 $c/M = 0.1000$

Δx (PIECEWISE LINEARIZATION) = 0.1000
 Δt (RUNGE-KUTTA) = 0.0500

$x(0) = 1.0000 \quad \dot{x}(0) = 0.0000$

— RUNGE-KUTTA
○ PIECEWISE LINEARIZATION

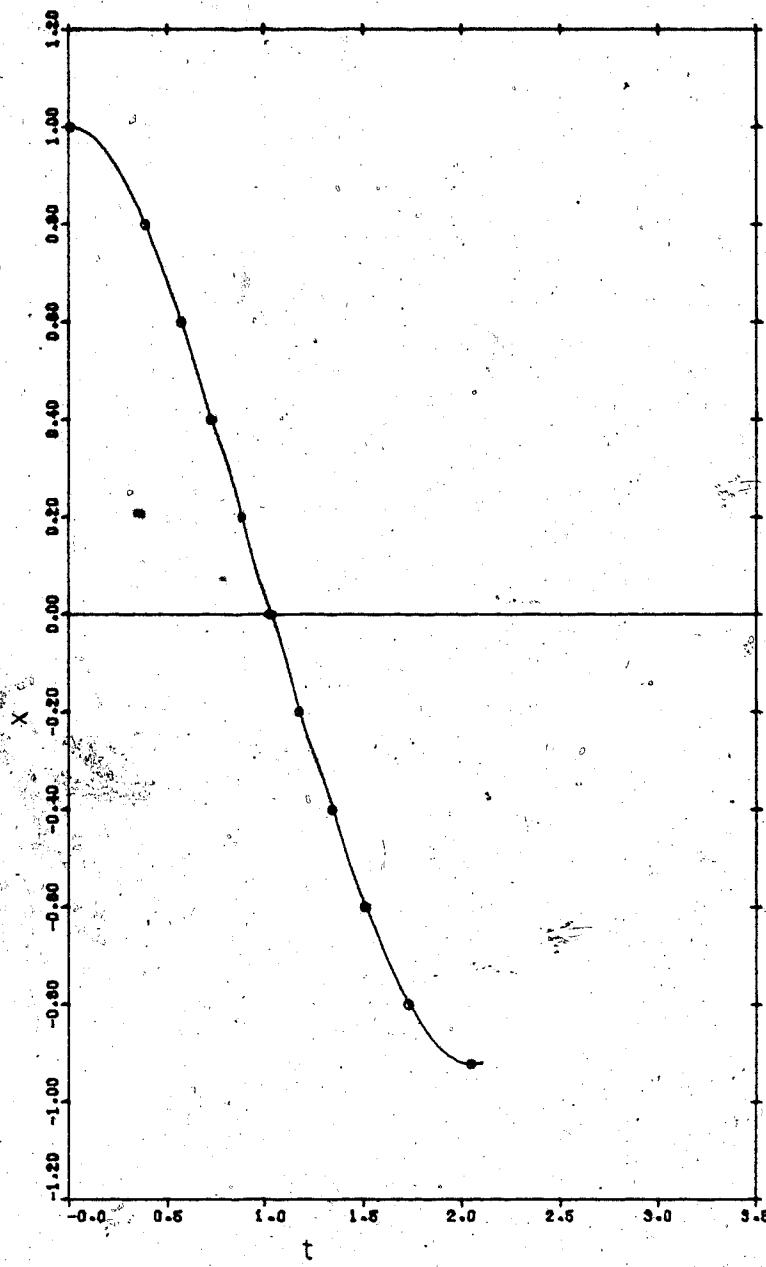


FIGURE 3.6 - SOLUTIONS TO HARD SPRING EQUATION
(LINEAR DAMPING) FOR $a = 1.0$, $b = 0.15$, $\frac{c}{m} = 0.25$,
 $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5].)

$a = 1.0000$ $b = 0.1500$
 $c/M = 0.2500$
 Δx (PIECEWISE
LINEARIZATION) = 0.1000
 Δt (RUNGE-KUTTA) = 0.0500
 $x(0) = 1.0000$ $\dot{x}(0) = 0.0000$

 — RUNGE-KUTTA
 ○ PIECEWISE LINEARIZATION

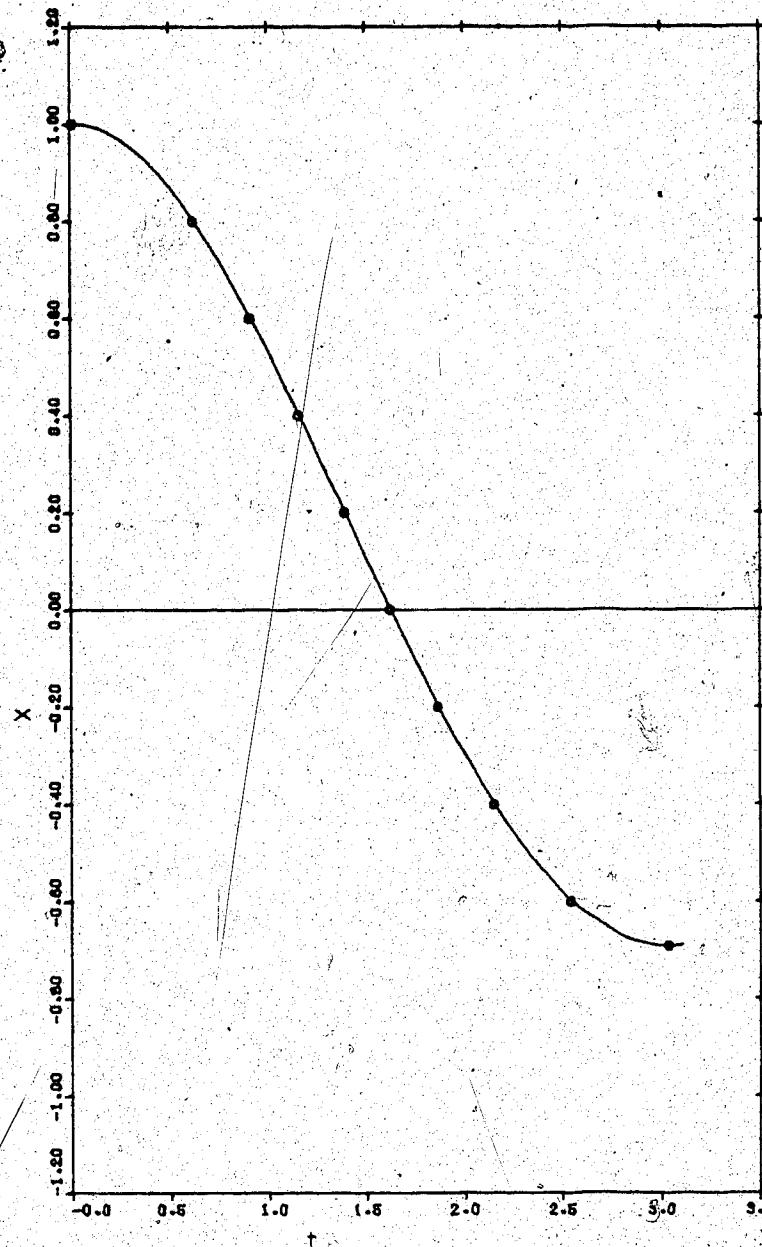


FIGURE 3.7 - SOLUTIONS TO HARD SPRING EQUATION
FOR a = 1.0, b = 2.0, $\frac{c}{m} = 0.25$,
 $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

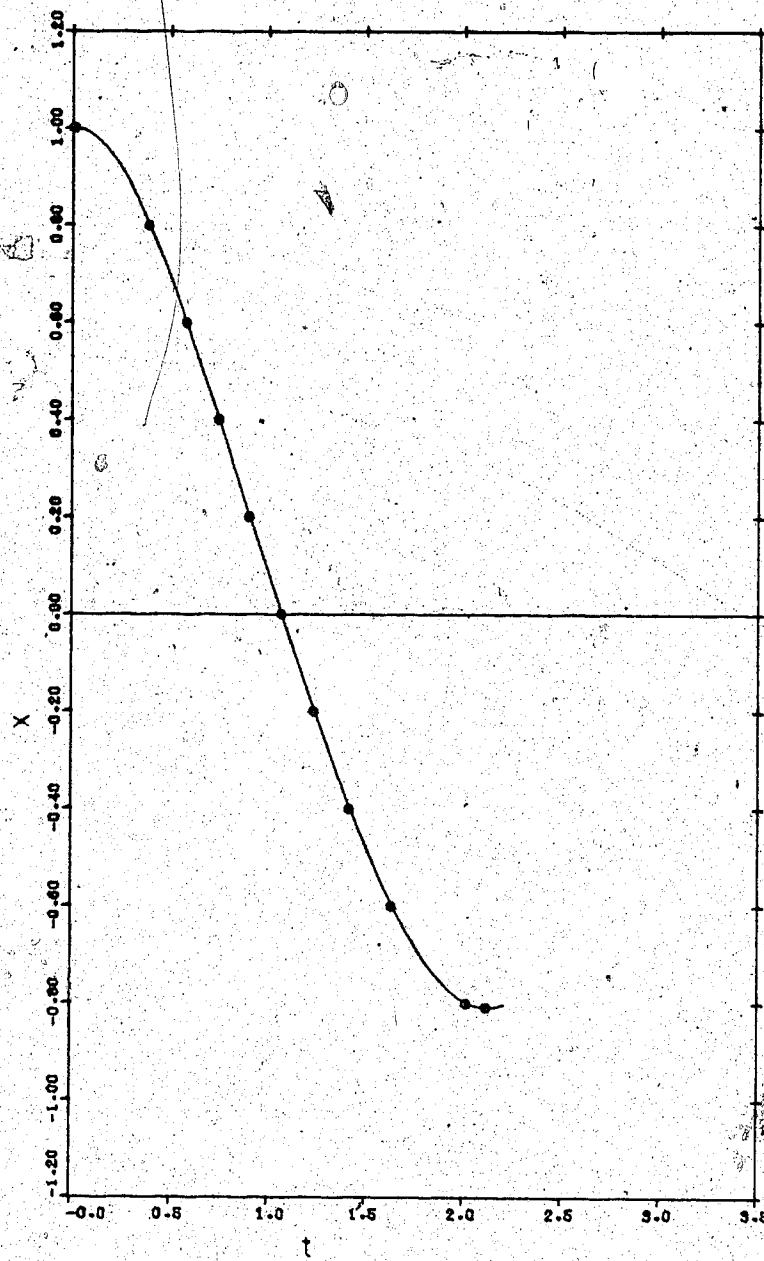
(N.B.: Runge-Kutta solution uses [5].)

$a = 1.0000$ $b = 2.0000$
 $c/M = 0.2500$

Δx (PIECEWISE
LINEARIZATION) = 0.1000
 Δt (RUNGE-KUTTA) = 0.0500

$x(0) = 1.0000$ $\dot{x}(0) = 0.0000$

— RUNGE-KUTTA
○ PIECEWISE LINEARIZATION



various parameters like x_0^* , as well as the input parameters and call to DVERK [5]. The results were obtained by running the program on the Amdahl 480V/8 [26] described earlier.

As a first trial, to see if the program functioned, it was decided to check if the solution would work for the simple case of $\frac{C}{M}$ being zero for the sets of values for a and b as described earlier in the chapter, and X_0 being taken to be 1.0. The results of this are seen in Figures 3.2 and 3.3, using a value of Δx of 0.1, and a time increment for the Runge-Kutta solution [5] of 0.05, the sizes being chosen arbitrarily, but still sufficiently large to allow any problems concerning stability arise.

From Figures 3.2 and 3.3, it is obvious that the program works for the undamped case, as the correspondence between the two solutions is good, and there appears to be little problem in the piecewise linear method bottoming out at the first trough.

Now for solving the equation using non-zero values for $\frac{C}{M}$. The results for this can be seen in Figures 3.4 - 3.7. For each plot there seems to be a good correspondence between the results obtained for the piecewise linear solution and the Runge-Kutta procedure [5], which implies two things. One is that the method of solution using the piecewise linearization by chords gives a good agreement with the Runge-Kutta approximation [5] of this equation. The other is that for even a somewhat large increment of displacement, the piecewise linear method gives a good result, and that it is not necessary to go to a smaller value to get an accurate solution. This has an advantage as far as the economics of the runs are concerned.

As before, the program was stripped down to the essential

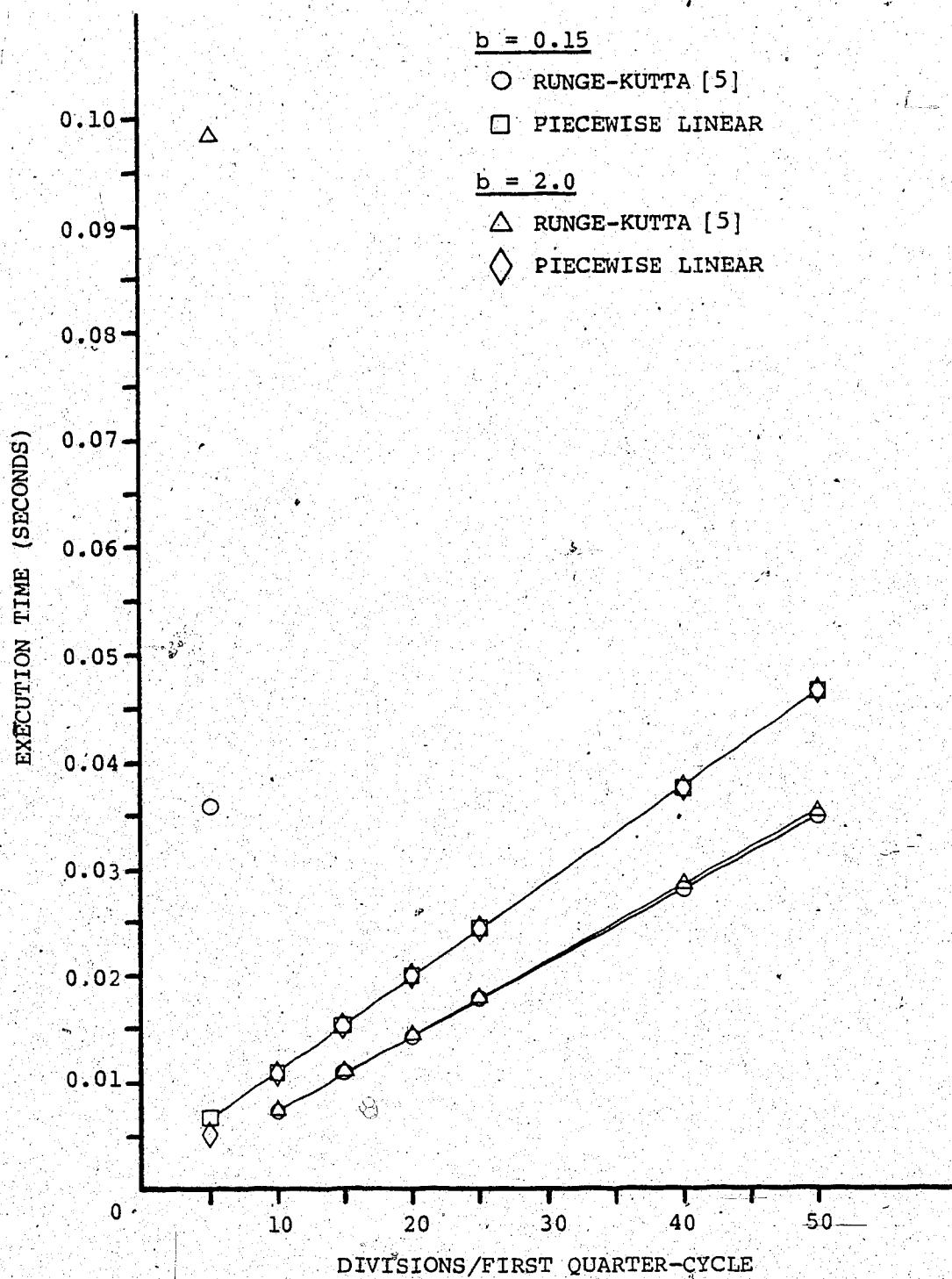


FIGURE 3.8 - EXECUTION TIMES FOR SOLUTIONS TO HARD SPRING EQUATION (LINEAR DAMPING) FOR
 $a = 1.0, \frac{c}{M} = 0.0, x(0) = 1.0, \dot{x}(0) = 0.0$ [27]

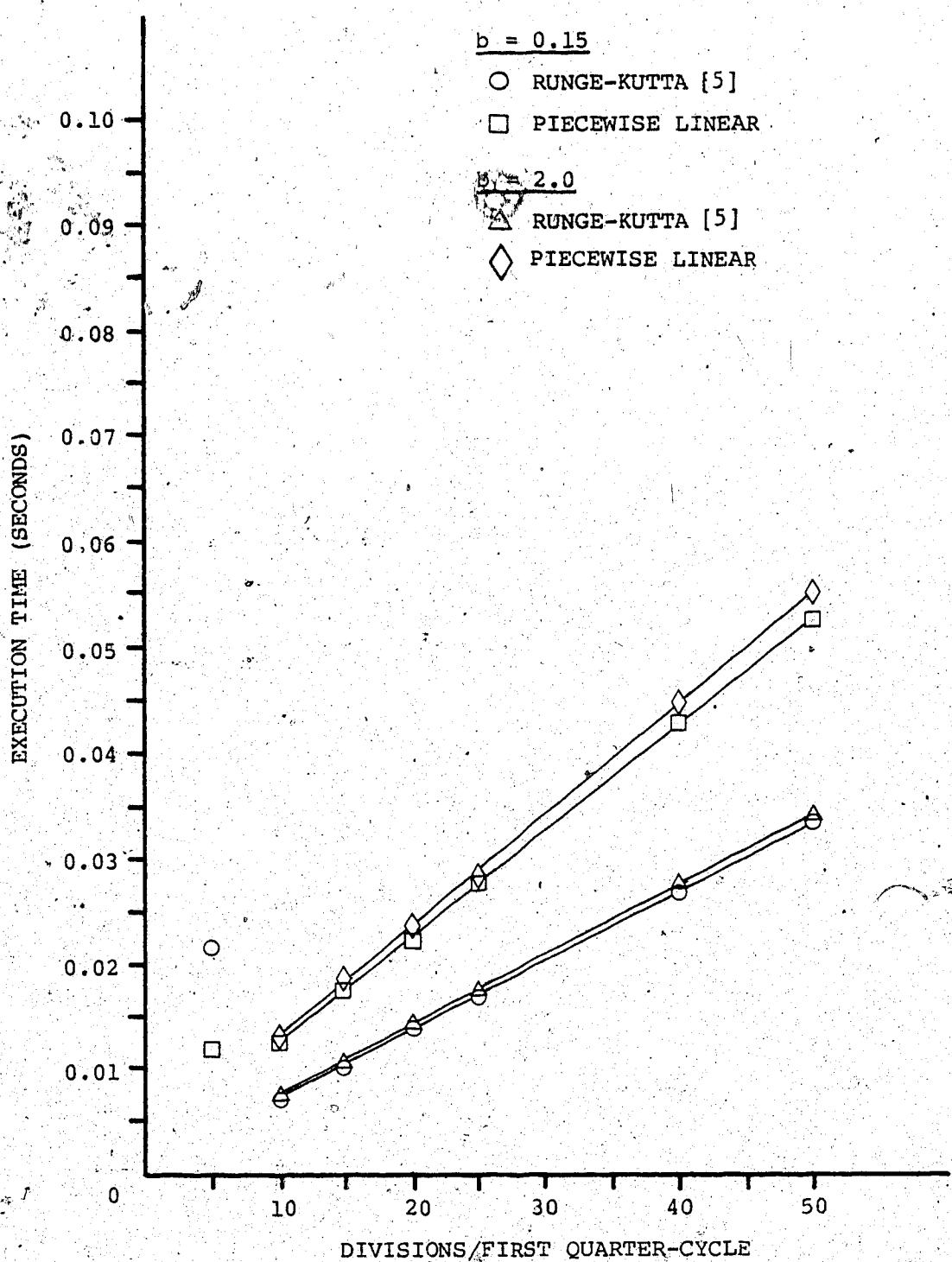


FIGURE 3.9 - EXECUTION TIMES FOR SOLUTIONS TO HARD SPRING EQUATION (LINEAR DAMPING) FOR
 $a = 1.0, \frac{c}{m} = 0.10, x(0) = 1.0, \dot{x}(0) = 0.0$ [27]

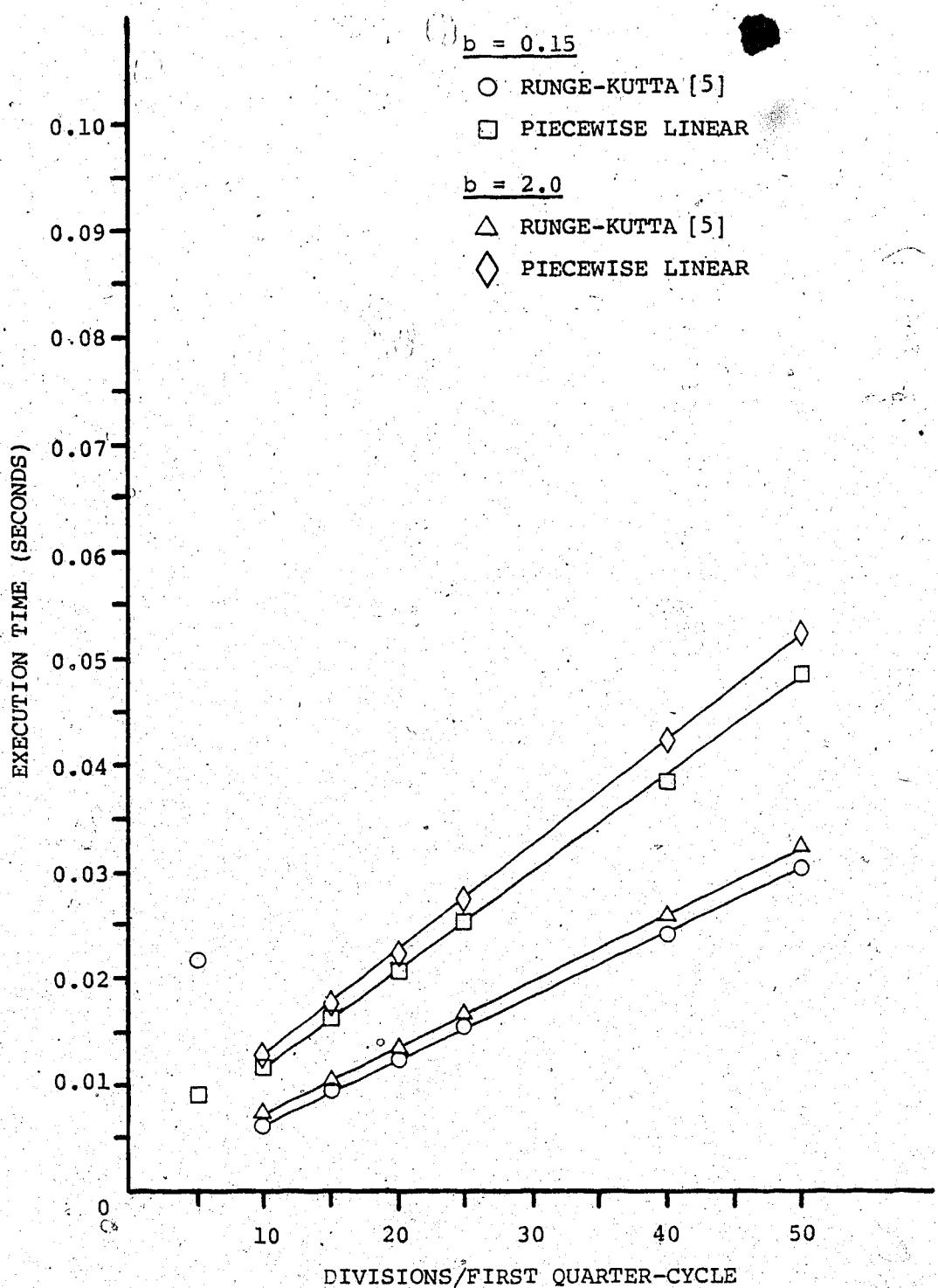


FIGURE 3.10 - EXECUTION TIMES FOR SOLUTIONS TO HARD SPRING EQUATION (LINEAR DAMPING) FOR
 $a = 1.0, \frac{C}{M} = 0.25, x(0) = 1.0, \dot{x}(0) = 0.0$ [27]

calculations and the execution times for each method of solutions obtained [27]. Again, it should be noted that these values were acquired with an older version of the plot program that was slower and that this was done on the V/7 version of the Amdahl 470 [25], [26]. The results can be seen in Figures 3.8 - 3.10.

Before continuing, an explanation is necessary concerning the timing results. As mentioned earlier, the piecewise method used an interval of displacement while the Runge-Kutta method [5] made use of time. For the purposes of determining correspondence between the two solutions, this would be sufficient, but this would not be valid for comparing the execution times, since Δx and Δt would scarcely provide any means for judgement between the two methods. However, the number of divisions (or overall iterations) for the first quarter-cycle would possibly be a benchmark. To an extent, this was done for the piecewise method, since the system initially had a known displacement, and Δx was chosen from that.

For the Runge-Kutta method [5], a similar approach can be used based upon the first quarter-period as calculated by the piecewise method. This timespan can be divided into the same number of intervals as was used by the first solution, and the execution time is obtained with the new value for Δt .

The timing program kept track of the number of points calculated by the piecewise solution (including the first trough, which generally falls inside an interval) based on Δx while the Runge-Kutta routine [5] calculates this same number of points, but using the new Δt as was just described. This should give a reasonable correspondence to the piecewise solution except that the Runge-Kutta method [5] will likely go past

the first trough as before.

One might expect a linear relationship between the execution time and the number of divisions of the first-quarter cycle. Since the Runge-Kutta solution [5] is from a packaged program, presumably it should be as efficient as possible, which would be a reasonable explanation as to why the execution times for it are lower as compared to the piecewise linear method. However, what is of note is what happens at 5 divisions.

For some of the runs, no values were obtained, as the execution time limit that had been set prior to the run had been exceeded. Attempts to restart the program by adding additional seconds to bring the total execution time limit to at least 15 seconds failed to give results [31]. Some clue may be available from those runs that used what may appear to be excessive run time values.

For runs such as for $a = 1.0$, $b = 0.15$, and $\frac{c}{M} = 0.25$, the points are far off the line. This may indicate that there could be a lower limit for which either solution would work (or be economical), as from what was obtained, one can see that the Runge-Kutta [5] took longer to finish at this point than the piecewise linear method, which was not expected.

IV. HARD SPRING EQUATION (NONLINEAR DAMPING)

A. Preliminary Comments

This situation is essentially the same as that examined in the previous chapter, with the exception that the damping is no longer linear, but is related to the square of the velocity. The solution for the linear segments is the same, but the means by which it is obtained are slightly different. The equation considered is:

$$x + \frac{q}{M} |\dot{x}|^2 \operatorname{sgn}(\dot{x}) + ax + bx^3 = 0, \quad (4.1.1)$$

where:

q = Nonlinear damping constant,

M, a, b = as used in Chapter III.

This equation describes the system shown in Figure 4.1 [28].

As in the case of equation (3.1.1), no exact solution seems to exist, and so the same Runge-Kutta solution [5] was used as a basis for comparison. The same values for a and b were used, while the values for $\frac{q}{M}$ were 0.0, 0.1, and 0.25, restricting the investigation to the under-damped case. The same initial conditions were used, with x_0 equal to 1.0.

B. Derivation of Piecewise Linear Solution

The derivation for the equation of the linear segments is exactly the same as was described in the previous chapters, using essentially the same method to find the necessary factors such as x_0^* and p_0^* . A slight modification in the procedure to determine the damping constant $\frac{c}{M}$

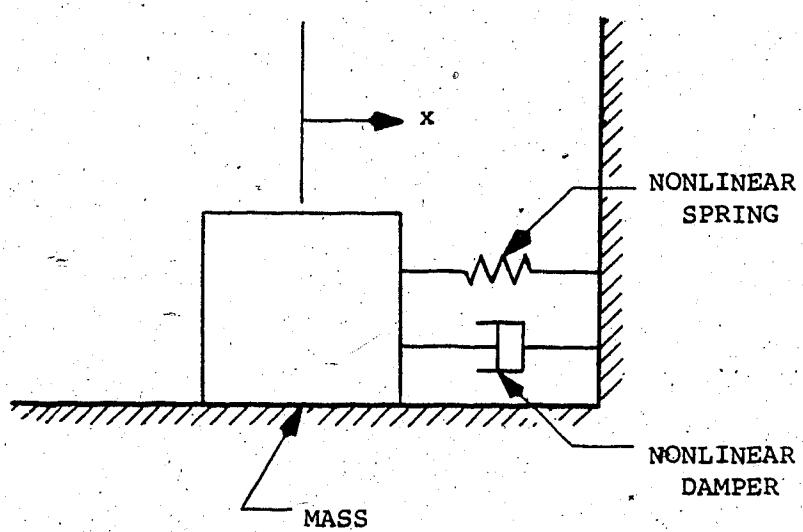


FIGURE 4.1 - MASS-SPRING SYSTEM (NONLINEAR DAMPING) [28]

is required as it changes in value from one segment to another. Since it is desired to obtain a solution whose form is similar to that of (1.2), the expression for the damping as given in (4.1.1) is replaced by an equivalent one resembling what was seen in (3.1.1), or, in other words, a linear approximation over a small increment of displacement

The starting point for this is part of the problem. One possibility involves an iteration within an iteration in order to obtain an average value of the velocity, and thus the damping force. This is done as follows.

At the beginning of motion, no damping is assumed, and the form of the equation is taken to be [19]:

$$\frac{d^2}{dt^2}(x - x_0^*) + \frac{k_0}{M}(x - x_0^*) = 0 .$$

From this, an end velocity can be found. This is averaged, and a damping coefficient is determined using:

$$c = \frac{q|\dot{x}_{AV}|^2 \operatorname{sgn}(\dot{x}_{AV})}{\dot{x}_{AV}} ,$$

\dot{x}_{AV} = Average segment velocity.

The iteration is done again, with the same initial conditions, but the equation now becomes [19], [29]:

$$\frac{d^2}{dt^2}(x - x_0) + \frac{c}{M}\dot{x} + \frac{k_0}{M}(x - x_0^*) = 0 .$$

Now the current velocity is calculated and compared with the previous

final value. If their difference is greater than a given tolerance, the iteration starts again, with a new damping coefficient calculated from the average segment velocity, based upon this new final speed. Once this tolerance is achieved, the end conditions become the initial conditions for the next segment, and the entire procedure starts again as before, except now for all subsequent segments, the new initial velocity is used each time.

Since the derivation is the same as before, the solution will essentially be identical to that of the previous chapter [19], [29] or:

$$x = x_0^* + e^{-n(t - t_0)} [A \cos p^*(t - t_0) + B \sin p^*(t - t_0)] \quad (4.2.1)$$

$$\dot{x} = e^{-n(t - t_0)} [(-nA + p^*B) \cos p^*(t - t_0) - (p^*A + nB) \sin p^*(t - t_0)] \quad (4.2.2)$$

where the coefficients were determined as before, and are:

$$A = x_0 - x_0^*$$

$$B = \frac{\dot{x}_0 + n(x_0 - x_0^*)}{p^*}$$

$$n = \frac{C}{2M}$$

The values for $t_1 - t_0$ were calculated using the Newton-Raphson method [30], and are subject to the conditions described in the previous chapter, including the procedure for finding the point at which the system bottoms out in the first trough.

C. Results

The equations for the piecewise linear solution and the required commands to use DVERK [5] (including parameters and the calling statement) were assembled into a computer program, and solution plots generated by running them on the Amdahl 470V/8 [26] mentioned earlier.

The size of the displacement increment used by the piecewise linear method was the same as the one used in the previous chapter ($\Delta x = 0.1$), as well as the same size of time increment for the Runge-Kutta method [5] ($\Delta t = 0.05$).

The results of the computer plot program that calculated both solutions can be seen in Figures 4.2 - 4.7. Once again, one can see that a good agreement was achieved, even when using a fairly large value for Δx , and it would seem from the output that the piecewise linear method successfully calculated the location of the first trough. It can be concluded from this that the parameter values and the size of the displacement interval were sufficient to give satisfactory results for the given conditions.

The program was stripped to the essential steps that contained only the calculations and the execution times were determined for it [27], which should give one a reasonable estimate as to the relative speeds (and hence, the costs) of the two methods. As in the previous chapter, the basis of judgement was the number of divisions of the first quarter-cycle. This was because the piecewise linear solution used an increment of displacement, while the Runge-Kutta method [5] required increments of time.

The values obtained are to be found on Figures 4.8 - 4.10, and were taken from tests conducted on the Amdahl 470V/7 [25], using an older and

FIGURE 4.2 - SOLUTIONS TO HARD SPRING EQUATION
(NONLINEAR DAMPING), FOR $a = 1.0$, $b = 0.15$, $M = 0.0$,
 $\Delta x(0) = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta method uses [5].)

$a = 1.0000$ $b = 0.1500$
 $q/M = 0.0000$

$\Delta x^{\text{PIECEWISE}} = 0.1000$
 $\Delta t \text{ (RUNGE-KUTTA)} = 0.0500$

$x(0) = 1.0000$ $\dot{x}(0) = 0.0000$

— RUNGE-KUTTA
○ PIECEWISE LINEARIZATION.

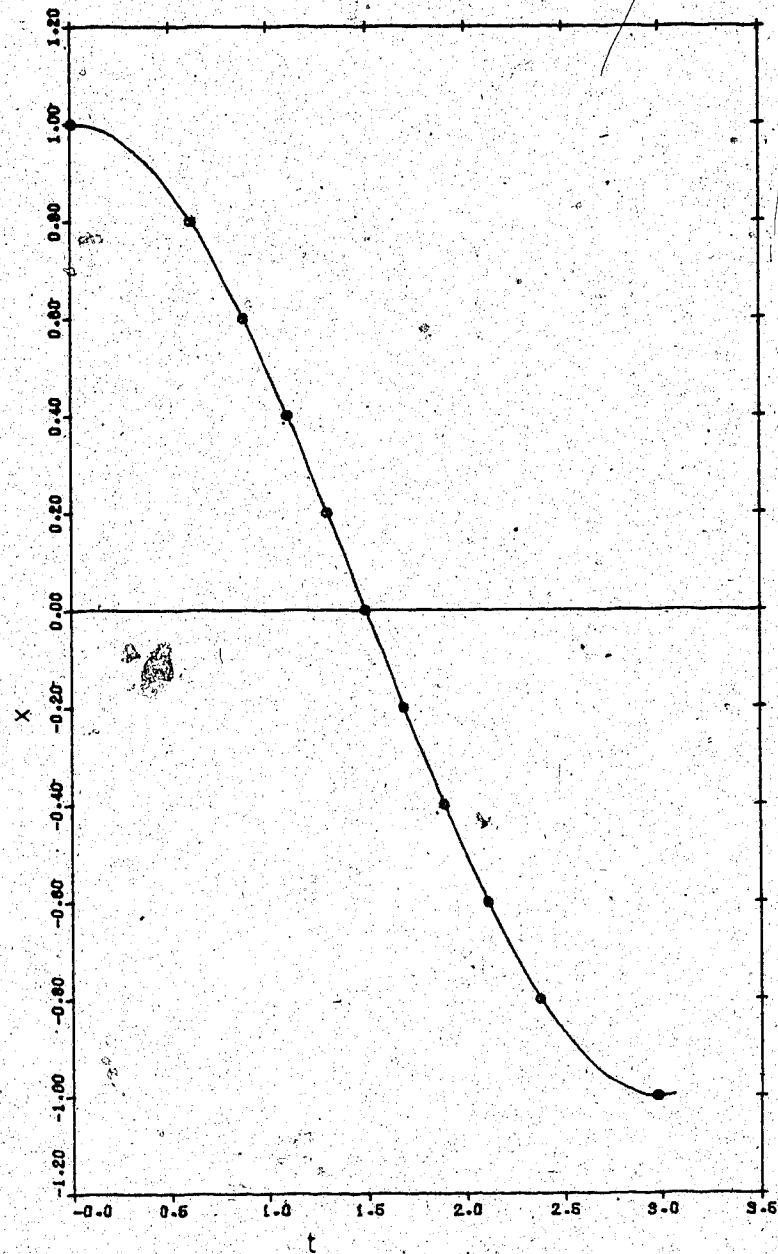


FIGURE 4.3 - SOLUTIONS TO HARD SPRING EQUATION
(NONLINEAR DAMPING) FOR $a = 1.0$, $b = 2.0$, $\frac{q}{M} = 0.0$,
 $\Delta x(0) = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta method uses [5].)

$a = 1.0000 \quad b = 2.0000$
 $q/M = 0.0000$

Δx (PIECEWISE LINEARIZATION) = 0.1000
 Δt (RUNGE-KUTTA) = 0.0500

$x(0) = 1.0000 \quad \dot{x}(0) = 0.0000$

— RUNGE-KUTTA
○ PIECEWISE LINEARIZATION

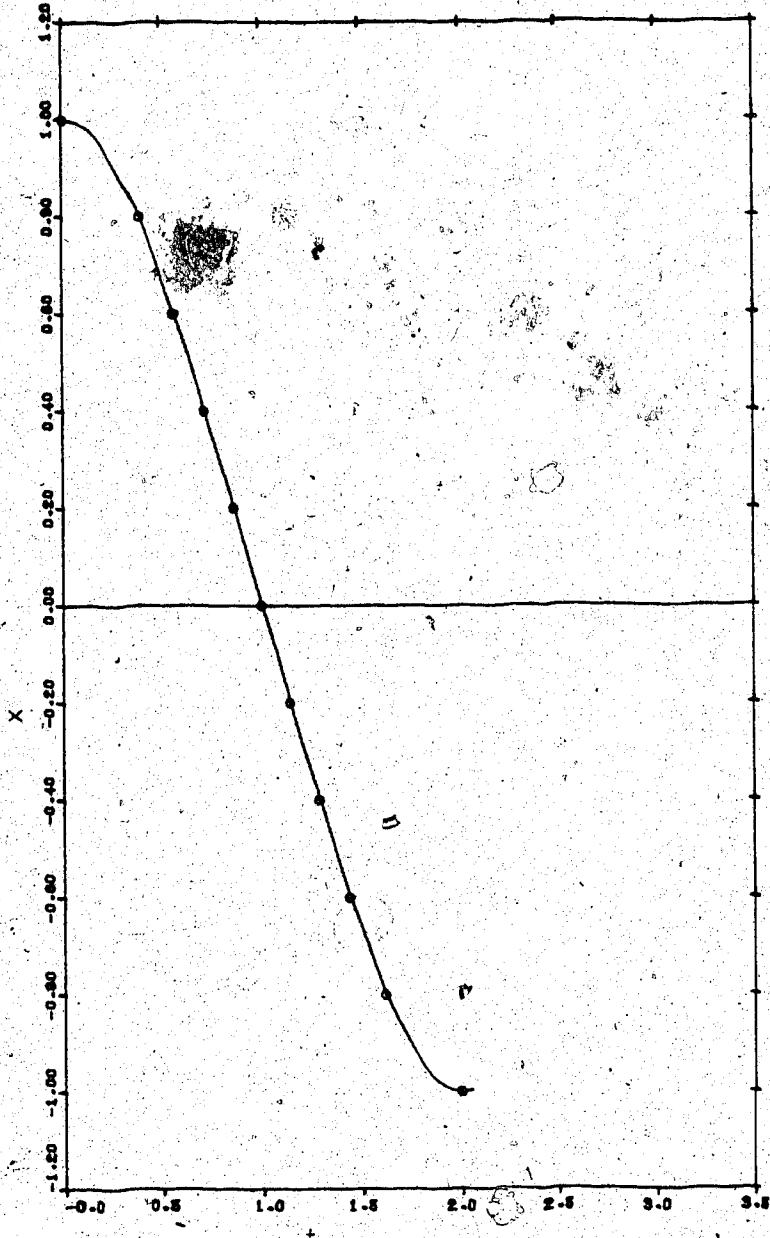


FIGURE 4.4 - SOLUTIONS TO HARD SPRING EQUATION
(NONLINEAR DAMPING) FOR $a = 1.0$, $b = 0.15$, $\Omega_0 = 0.10$,
 $\Delta x(0) = 0.10$, $\dot{x}(0) = 0.05$, $x(0) = 1.0$, $\ddot{x}(0) = 0.0$

(N.B.: Runge-Kutta method uses [5].)

$$a = 1.0000 \quad b = 0.1500$$

$$q/M = 0.1000$$

$$\Delta x (\text{PIECEWISE LINEARIZATION}) = 0.1000$$

$$\Delta t (\text{RUNGE-KUTTA}) = 0.0500$$

$$x(0) = 1.0000 \quad \dot{x}(0) = 0.0000$$

— RUNGE-KUTTA
○ PIECEWISE LINEARIZATION

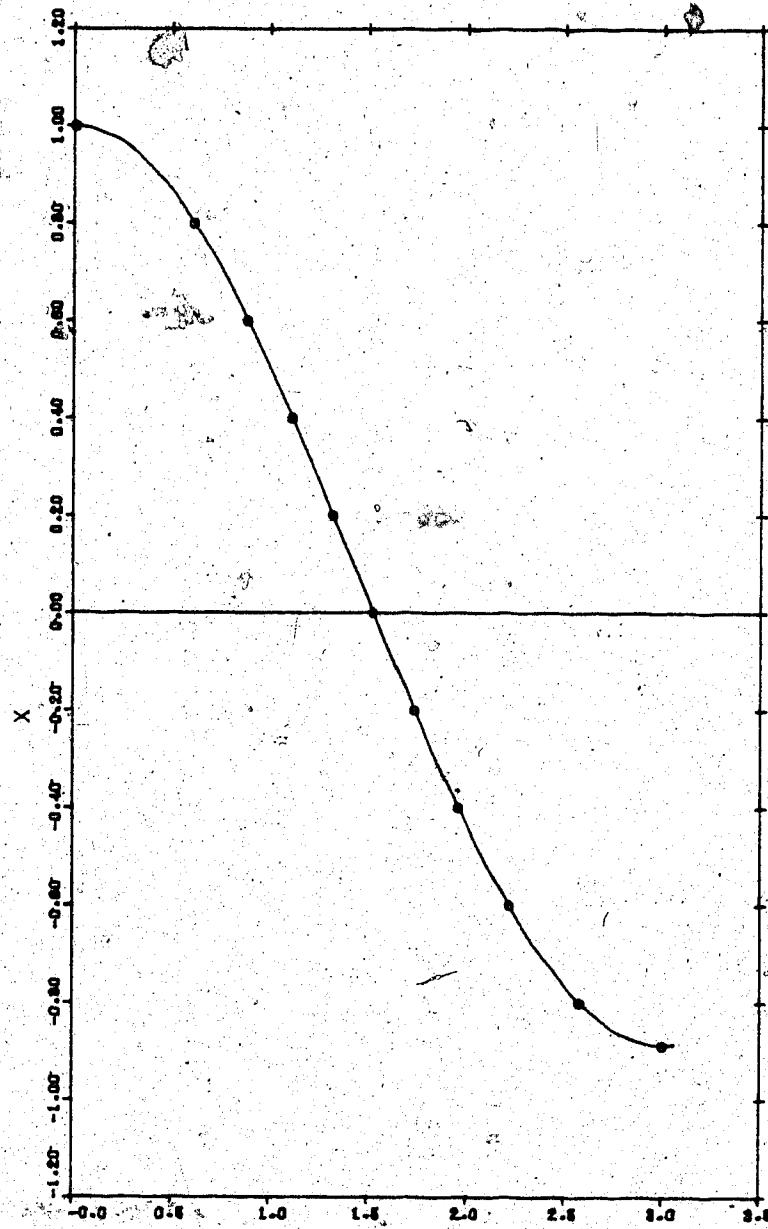


FIGURE 4.5 - SOLUTIONS TO HARD SPRING EQUATION
(NONLINEAR DAMPING) FOR $a = 1.0$, $b = 2.0$, $\frac{g}{M} = 0.10$,
 $\Delta x(0) = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta method uses [5].)

$a = 1.0000 \quad b = 2.0000$
 $q/M = 0.1000$

Δx (PIECEWISE LINEARIZATION) = 0.1000
 Δt (RUNGE-KUTTA) = 0.0500

$x(0) = 1.0000 \quad \dot{x}(0) = 0.0000$

— RUNGE-KUTTA
○ PIECEWISE LINEARIZATION

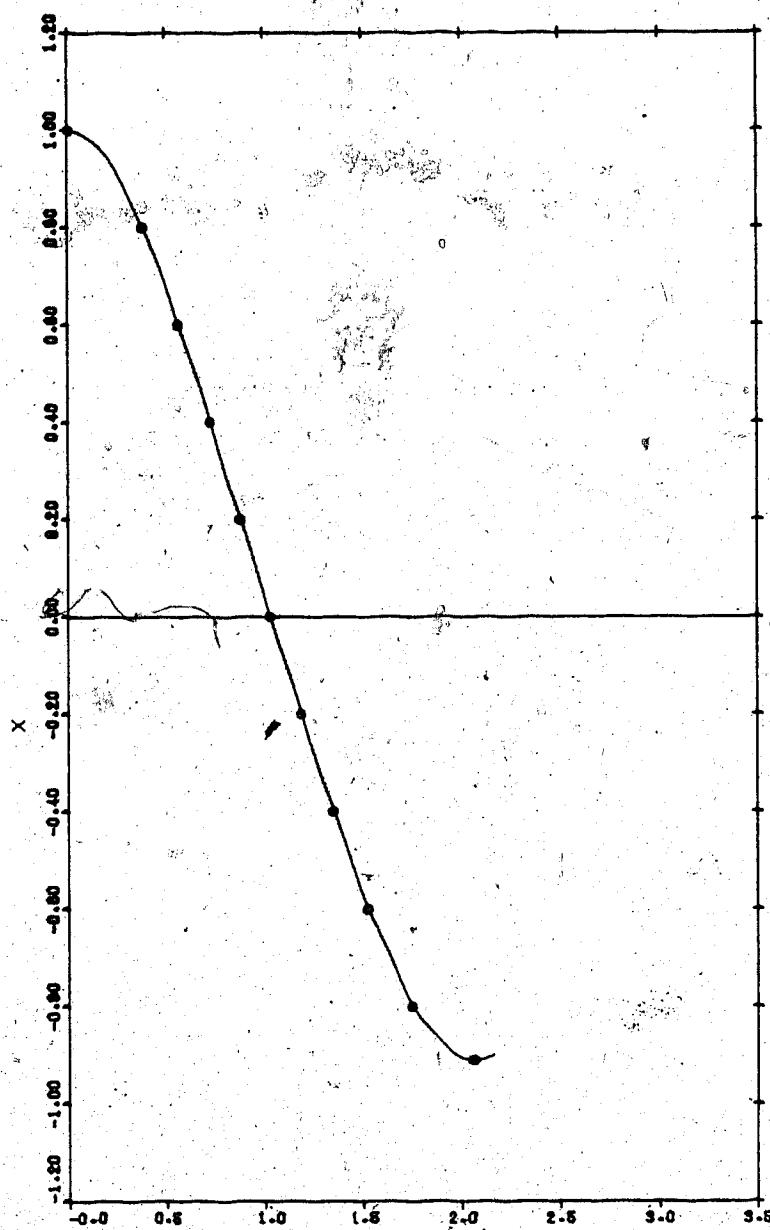


FIGURE 4.6 - SOLUTIONS TO HARD SPRING EQUATION
(NONLINEAR DAMPING) FOR $a = 1.0$, $b = 0.15$, $M = 0.25$,
 $\Delta x(0) = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta method uses [5].)

$a = 1.0000 \quad b = 0.1500$
 $q/M = 0.2500$

Δx (PIECEWISE LINEARIZATION) = 0.1000
 Δt (RUNGE-KUTTA) = 0.0500

$x(0) = 1.0000 \quad \dot{x}(0) = 0.0000$

— RUNGE-KUTTA
○ PIECEWISE LINEARIZATION

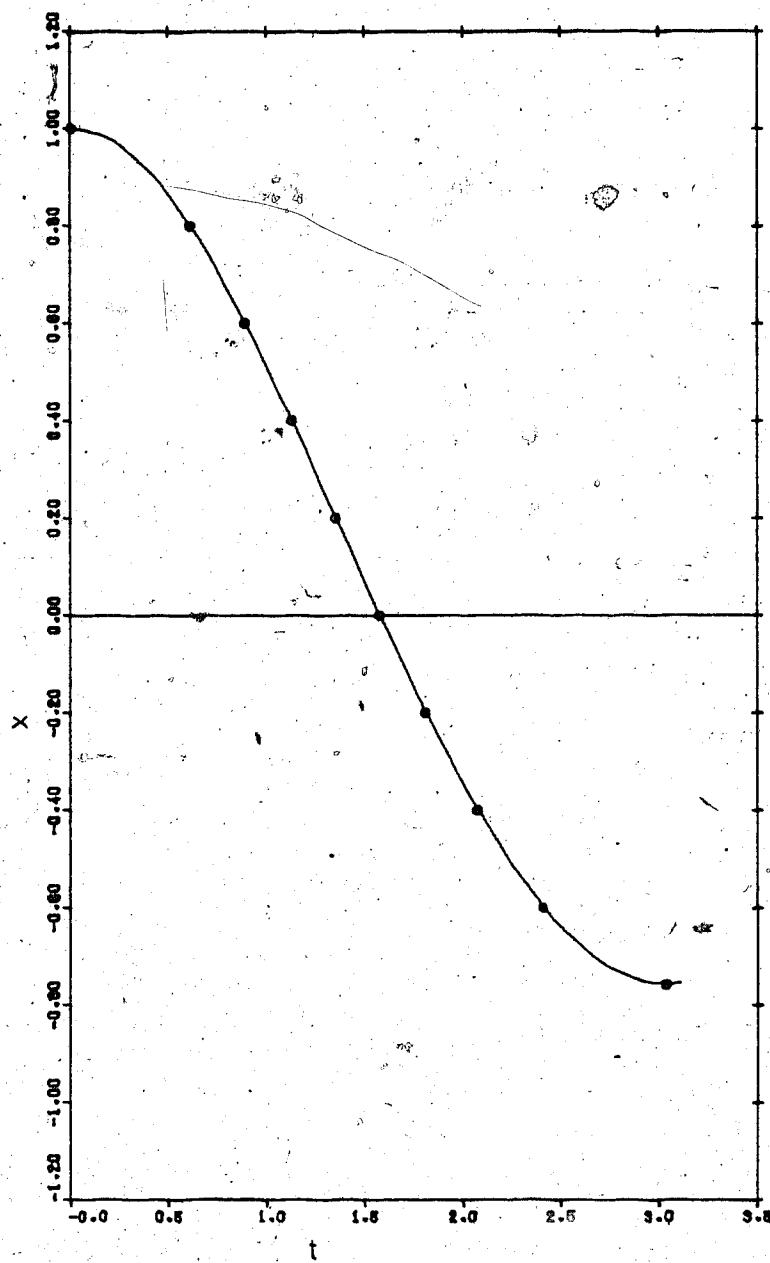


FIGURE 4.7 - SOLUTIONS TO HARD SPRING EQUATION
(NONLINEAR DAMPING) FOR $a = 1.0$, $b = 2.0$, $\frac{Q}{M} = 0.25$,
 $\Delta x(0) = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

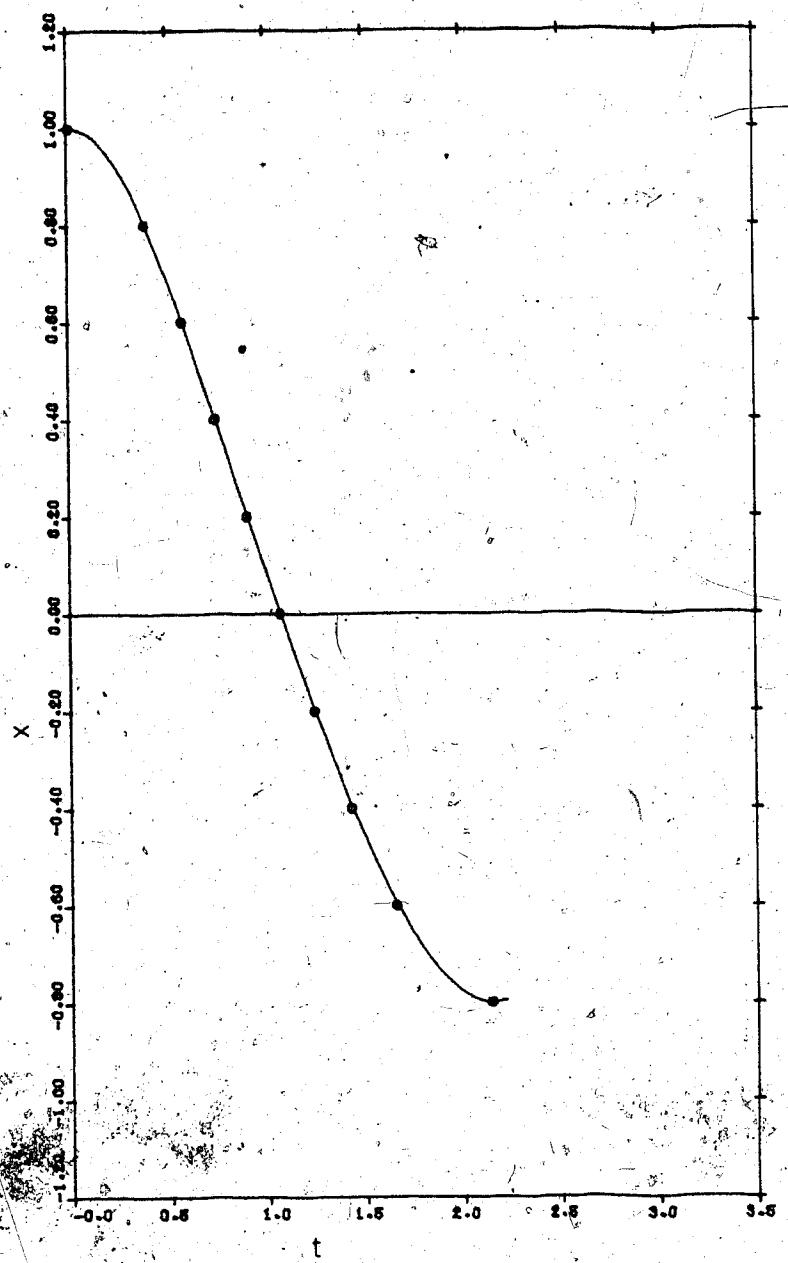
(N.B.: Runge-Kutta method uses [5].)

$a = 1.0000 \quad b = 2.0000$
 $q/M = 0.2500$

Δx (PIECEWISE LINEARIZATION) = 0.1000
 Δt (RUNGE-KUTTA) = 0.0500

$x(0) = 1.0000 \quad \dot{x}(0) = 0.0000$

— RUNGE-KUTTA
○ PIECEWISE LINEARIZATION



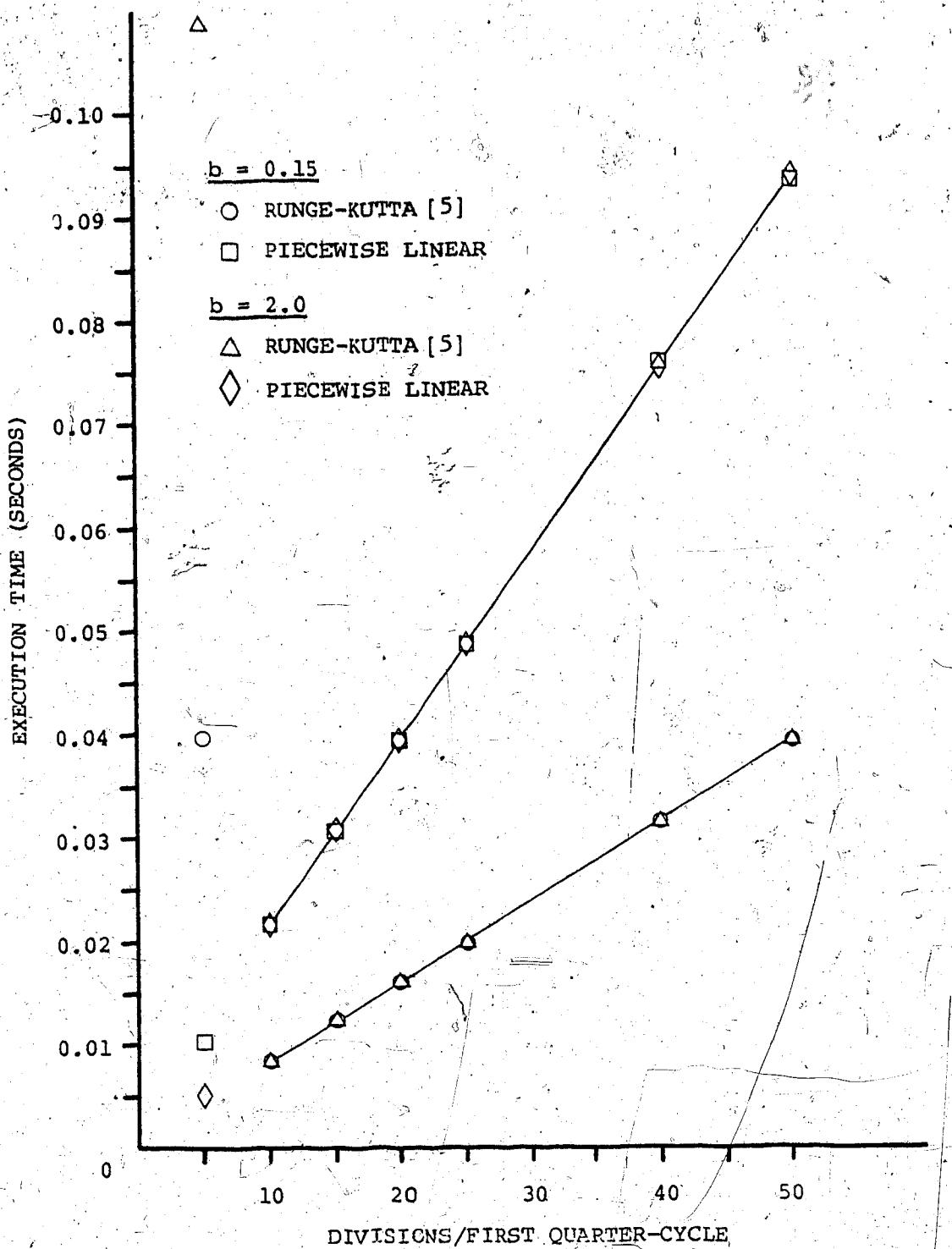


FIGURE 4.8 - EXECUTION TIMES FOR SOLUTIONS TO HARD SPRING EQUATION (NONLINEAR DAMPING) FOR $a = 1.0$, $\frac{g}{M} = 0.0$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$. [27]

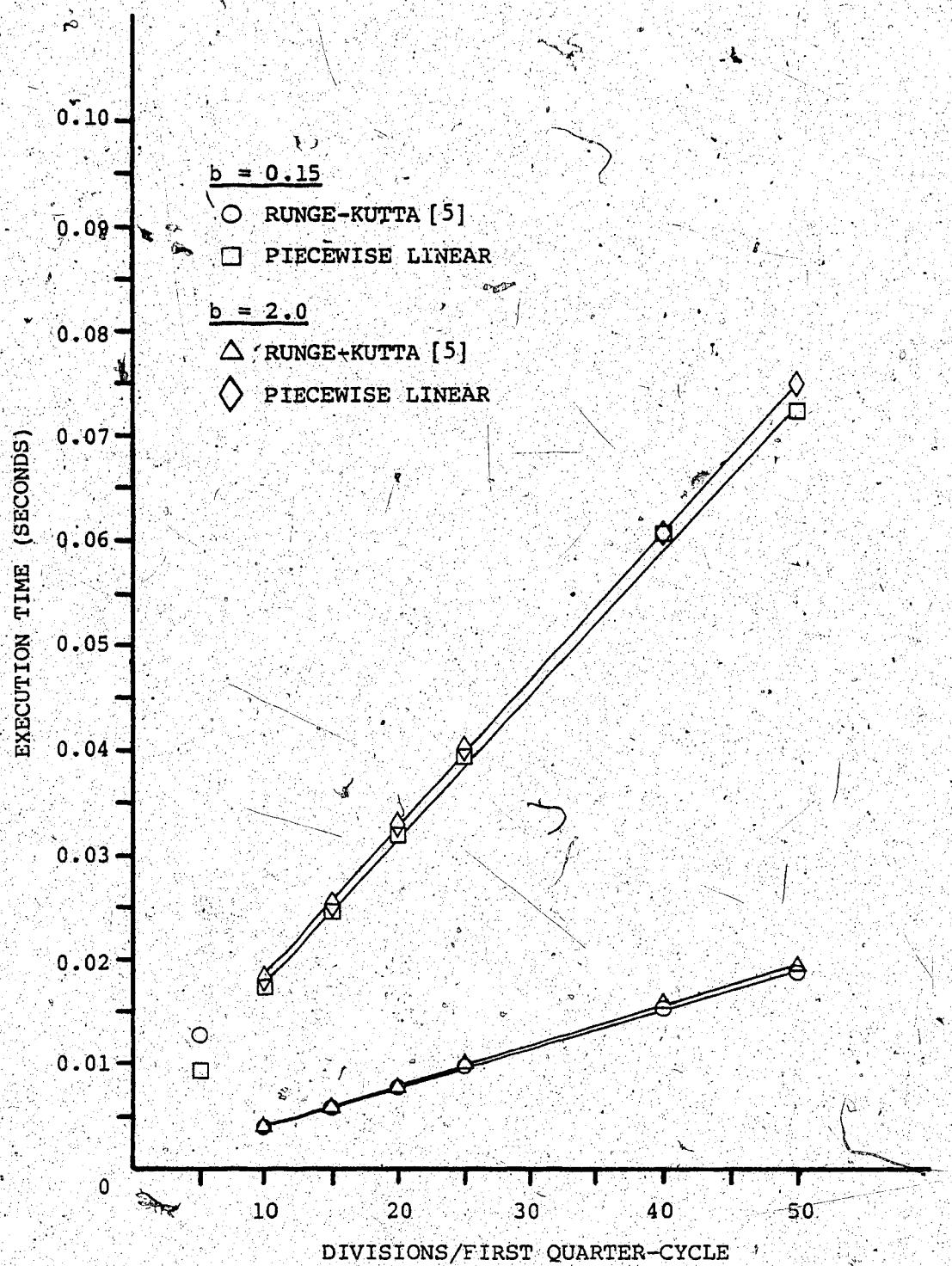


FIGURE 4.9 - EXECUTION TIMES FOR SOLUTIONS TO HARD SPRING EQUATION (NONLINEAR DAMPING) FOR $a = 1.0$, $\frac{q}{M} = 0.10$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$ [27]

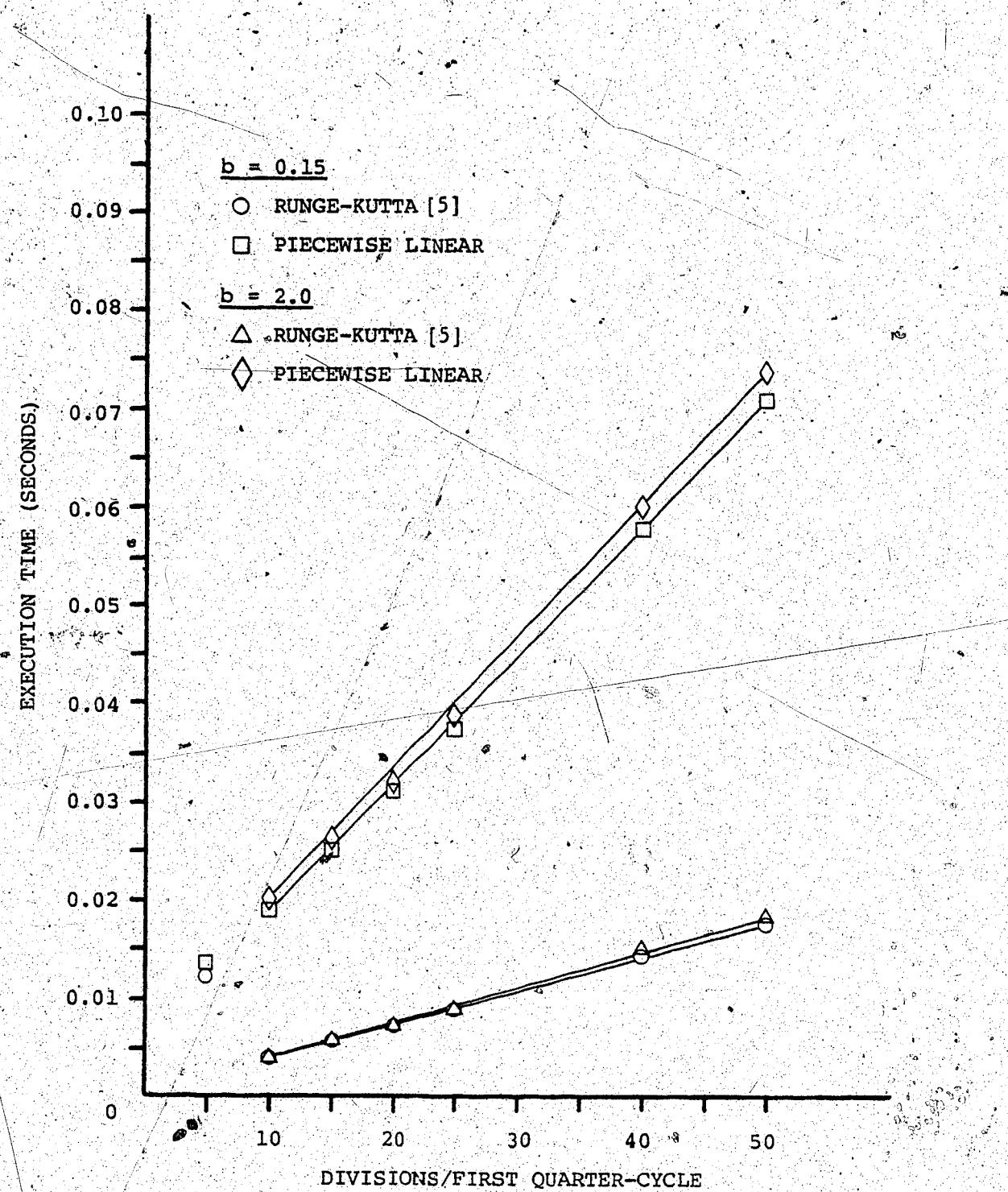


FIGURE 4.10 - EXECUTION TIMES FOR SOLUTIONS TO HARD SPRING EQUATION (NONLINEAR DAMPING) FOR $a = 1.0$,
 $\frac{q}{M} = 0.25$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$ [27]

slower version of the program.

Once again, it was found that the time required was linear with respect to the number of divisions per first quarter-cycle. As before, for 5 divisions of the first quarter-cycle, problems were experienced in obtaining a reasonable execution time for either result, depending upon the value of b and $\frac{q}{M}$ (since an increase in either or both caused the program to exceed its pre-set execution time limit [31]). This leads to the same conclusion as before that there may be a lower limit to the number of points that can be used for either method to be effective or economical. One of the things that was surprising was the lowering of the execution time for the piecewise method for $a = 1.0$, $b = 2.0$, and $\frac{q}{M} = 0.0$. This could be regarded as coincidence, unless there was some factor in the computer system itself that may have led to this which is not immediately apparent.

V. VAN DER POL EQUATION

A. Preliminary Comments

The methods of solution described in the previous two chapters gave stable results since, when the damping factors were varied over a large range for the underdamped case, no erratic behaviour or improbable wave forms were encountered. In this chapter, however, an equation will be examined that is different in that the term representing its resisting force varies with respect to both the displacement and the velocity, its solution curve varies radically as a parameter is varied over a wide range, and its phase trajectory exhibits a limit cycle. This is Van der Pol's equation [32]:

$$\ddot{x} + \mu(x^2 - 1)\dot{x} + x = 0. \quad (5.1.1)$$

The solutions to the linear equations are essentially the same as were found in the previous chapters, except that instead of being restricted to only the underdamped situation, the critically damped and overdamped cases are solved for as well, owing to the nature of the solution to (5.1.1).

Owing to the wide variation in behaviour that this equation can exhibit, the initial conditions will be the same as had been previously used with the exception that x_0 will vary in value to allow investigation inside, on, and outside the limit cycle. The parameter μ will have the values 0.25, 1.0, 2.0, and 5.0 so that the variations in the solution curves can be examined.

While the previous systems had nonlinear restoring forces, (5.1.1)

does not, and so, this will mean that many of the calculations that were needed in order to set up the solution to the linear equation are not needed in this situation. This also means that an interval of time can be used instead of displacement with resulting simplification in the calculations.

Since an exact solution does not seem to exist, the same Runge-Kutta solution [5] that had been used in the previous two chapters will be the benchmark against which the piecewise linearization will be compared.

B. Derivation of Piecewise Linear Solution

Equation (5.1.1) can be approximated over a small increment of time by the equation [29]:

$$\ddot{x} + 2n\dot{x} + p^2 x = 0 \quad (5.2.1)$$

where:

$$n \approx \frac{u(x^2 - 1)}{2} \quad (5.2.2)$$

$$p^2 = 1.0$$

The solutions of (5.2.1) can be found in texts (for example, Timoshenko, et. al. [29]), and are as follows.

For $p^2 - n^2 > 0$ [35], [36]:

$$x = e^{-nt} \left[x_0 \cos p^* t + \frac{nx_0 + \dot{x}_0}{p^*} \sin p^* t \right] \quad (5.2.3)$$

$$\dot{x} = e^{-nt} \{ \dot{x}_0 \cos p^* t + \left[\frac{-n(nx_0 + \dot{x}_0)}{p^*} - p^* x_0 \right] \sin p^* t \}, \quad (5.2.4)$$

$$p^* = \sqrt{p^2 - n^2}$$

For $p^2 - n^2 = 0$ [36], [37]:

$$x = e^{-pt} [x_0 + (nx_0 + \dot{x}_0)t], \quad (5.2.5)$$

$$\dot{x} = e^{-pt} [-px_0 + (nx_0 + \dot{x}_0)(1 - pt)]. \quad (5.2.6)$$

For $p^2 - n^2 < 0$ [36], [37]:

$$x = e^{-nt} [x_0 \cosh(p^{**}t) + \left(\frac{nx_0 + \dot{x}_0}{p^{**}} \right) \sinh(p^{**}t)], \quad (5.2.7)$$

$$\dot{x} = e^{-nt} \{ \dot{x}_0 \cosh(p^{**}t) + \left[\frac{-n(nx_0 + \dot{x}_0)}{p^{**}} + p^{**}x_0 \right] \sinh(p^{**}t) \}. \quad (5.2.8)$$

$$p^{**} = \sqrt{n^2 - p^2}$$

The major problem here is how to estimate n . Assume at the beginning of a segment some initial conditions of x_0 and \dot{x}_0 , and a time interval of Δt_p . An initial guess would use x_0 for x in (5.2.2) and an initial estimate of the change in displacement is calculated (which will

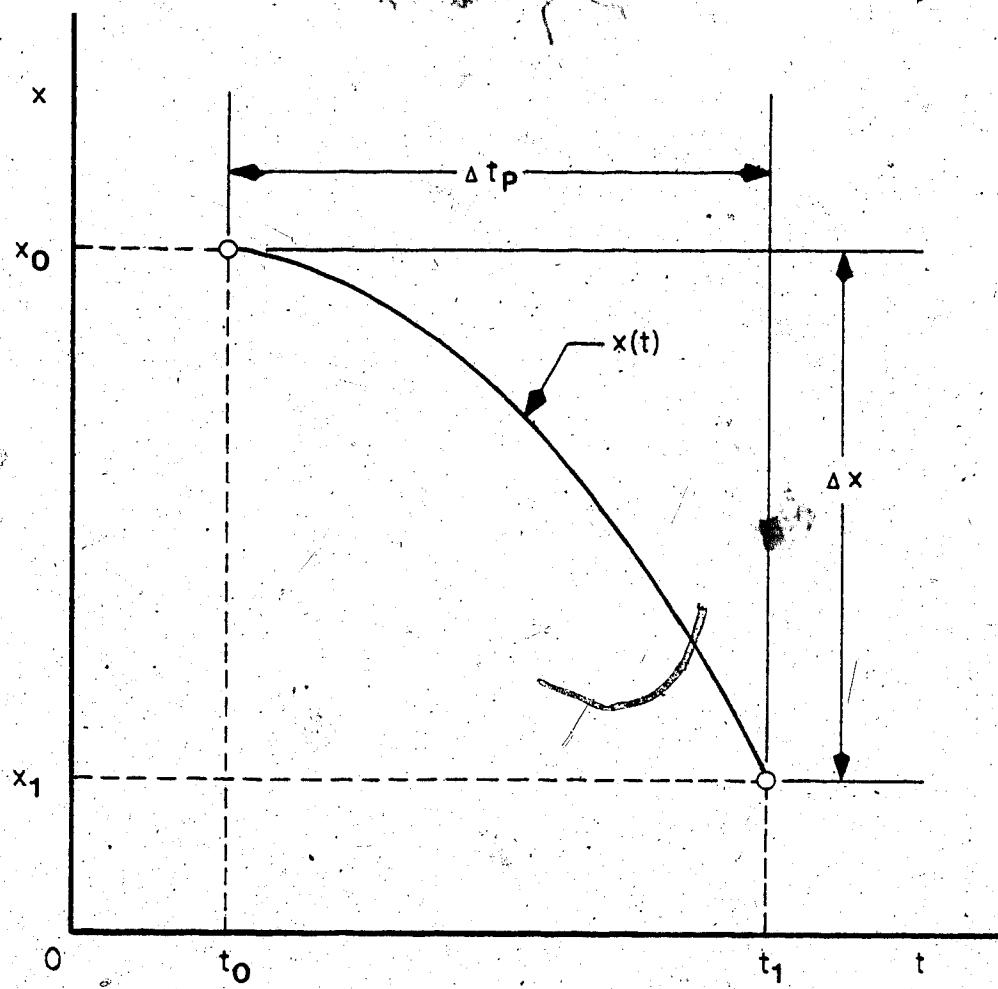


FIGURE 5.1 - TYPICAL INTERVAL LAYOUT (VAN DER POL EQUATION)

be explained further on). The new displacement x_1 is $x_0 + \Delta x$, and this is compared to a previously calculated value of x_1 (x_{PREV}). Should the difference between x_1 and x_{PREV} be very small, x_1 is the final displacement at the end of Δt_p . If not, x_1 becomes x_{PREV} , x now is $x_0 + \Delta x/2$, and a new n is found, as well as new values for Δx and x . Figure 5.1 should clarify matters.

This is how the solution is carried out in general for each interval. At the beginning of the iteration,

$$t_0 = \text{Initial time},$$

$$t_0 + \Delta t_p = \text{Final time} \\ = t_1,$$

$$x_0 = \text{Initial displacement}$$

$$x_0 + \Delta x = \text{Final displacement} \\ = x_1,$$

$$\dot{x}_0 = \text{Initial velocity},$$

$$\dot{x}_1 = \text{Final velocity}.$$

Once the displacement approaches either a peak or trough, a variation in this pattern is introduced to find the time and displacement at these points. (See Figure 5.2 for details.) The procedure is as follows. Starting at x_0 and t_0 , x_1 is found as before. The value of $\text{sgn}(\dot{x}_1)$ is calculated, and if found to be different from $\text{sgn}(\dot{x}_0)$, Δt_1 (the time taken to reach x_{11}) is found by determining the time interval needed for \dot{x} to reach zero using a Newton-Raphson method [30], starting at x_0 . This assumes that \dot{x} not zero at the end of a complete segment. Should it be, this section is circumvented.

The value for x_{11} is found by iteration much like the bottom of the

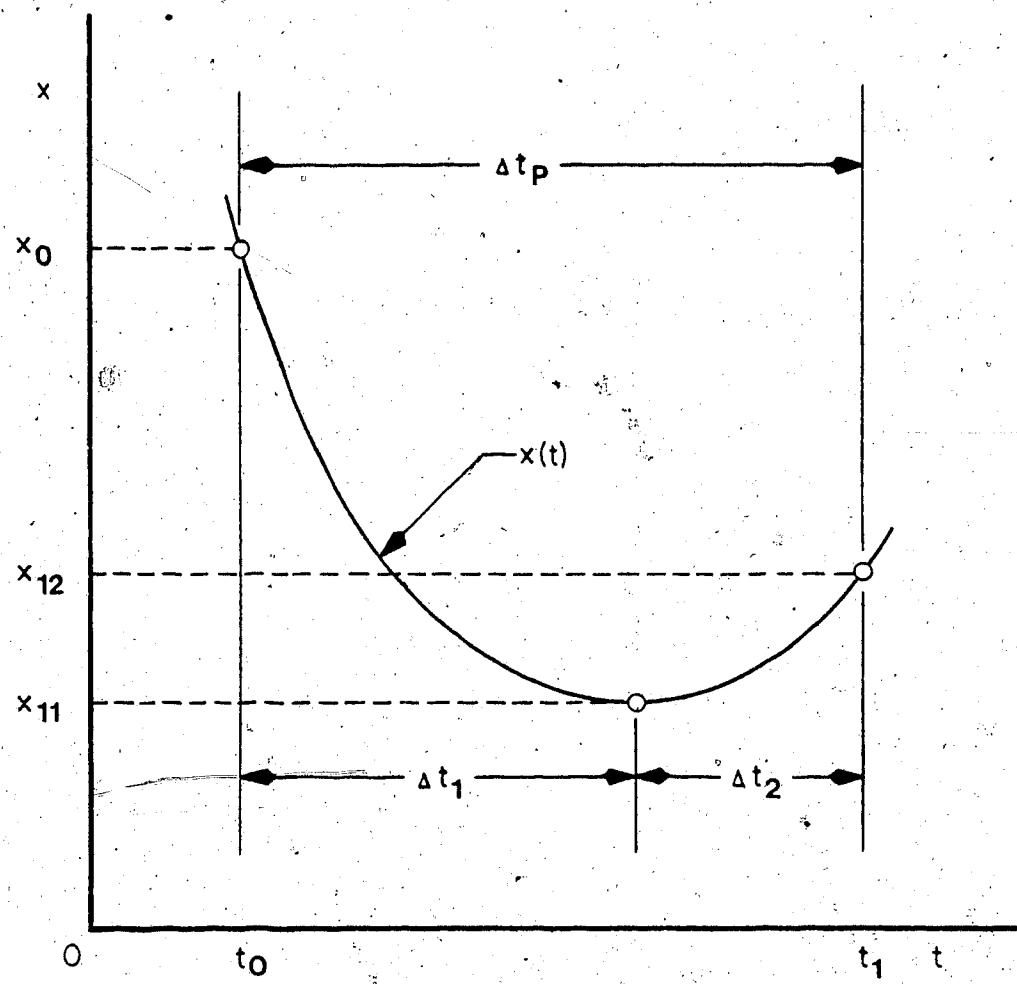


FIGURE 5.2 - LAYOUT OF PEAK OR TROUGH (VAN DER POL EQUATION)

first trough was determined in the previous two chapters. The estimated value is x_F , with the first guess being x_{12} . Then, using Δt_1 , a change in displacement is found (a new Δx). The value of $(x_0 + \Delta x) - x_F$ is determined, and if very small, x_{11} is equal to $x_0 + \Delta x$. If not, $x_0 + \Delta x$ becomes x_F and the iteration starts again.

This completes the first half-cycle.

Since the peak or trough is most likely not going to occur at the end of a time interval, and it would be convenient to find the displacement at the end of Δt_p (in order to continue afterwards with even time intervals until the next peak or trough), x_{12} is found based upon Δt_p , with the remainder of the half-cycle as was previously done. The displacement-time curve and phase-plane diagram can now be obtained.

It is necessary to go through this procedure as one never knows in advance how long the system takes to reach the limit cycle, since it depends upon the value of $p^2 - n^2$ as well as the initial conditions. So, for convenience, the number of half-cycles that had been calculated will be monitored and the calculation stops once the required number had been achieved.

When the displacement approaches a peak or trough, Δt_1 is determined using the Newton-Raphson method [30]:

$$(\Delta t)_{i+1} = (\Delta t)_i - \frac{\dot{x}[(\Delta t)_i]}{\ddot{x}[(\Delta t)_i]}$$

where $\dot{x}[(\Delta t)_i]$ is the expression for the velocity (which particular one would depend upon what value $p^2 - n^2$ has), and $\ddot{x}[(\Delta t)_i]$ is the acceleration for that particular situation.

The following approximations to the velocities and accelerations are used, based upon Timoshenko, et. al. [34].

For $p^2 - n^2 > 0$ [35], [36]:

$$\dot{x}[(\Delta t)_i] = e^{-n(\Delta t)_i} \{ A \cos[p^*(\Delta t)_i] \\ + B \sin[p^*(\Delta t)_i] \}, \quad (5.2.9)$$

$$x[(\Delta t)_i] = e^{-n(\Delta t)_i} \{ (-nA + p^*B) \\ \cdot \cos[p^*(\Delta t)_i] \\ + (-p^*A - nB) \sin[p^*(\Delta t)_i] \}, \quad (5.2.10)$$

$$A = \dot{x}_0,$$

$$B = -\frac{n(nx_0 + \dot{x}_0)}{p^*} - p^*x_0.$$

For $p^2 - n^2 = 0$ [35], [36]:

$$\dot{x}[(\Delta t)_i] = e^{-p(\Delta t)_i} \{ -px_0 + (nx_0 + \dot{x}_0) \\ \cdot [1 - p(\Delta t)_i] \}, \quad (5.2.11)$$

$$x[(\Delta t)_i] = e^{-p(\Delta t)_i} \{ p^2x_0 - p(nx_0 + \dot{x}_0) \\ \cdot [2 - p(\Delta t)_i] \}. \quad (5.2.12)$$

For $p^2 - n^2 < 0$ [35], [36]:

~~$$\dot{x}[(\Delta t)_i] = e^{-n(\Delta t)_i} \{ C \cosh[p^{**}(\Delta t)_i] \\ + D \sinh[p^{**}(\Delta t)_i] \}, \quad (5.2.13)$$~~

$$\begin{aligned} x[(\Delta t)_1] &= e^{-n(\Delta t)} \{ (-nC + p^{**}D) \\ &\quad \cdot \cosh [p^{**}(\Delta t)_1] \\ &\quad + (p^{**}C - nD) \sinh [p^{**}(\Delta t)_1] \}, \end{aligned} \quad (5.2.14)$$

$$C = \dot{x}_0,$$

$$D = -\frac{n(nx_0 + \dot{x}_0)}{p^{**}} + p^{**}x_0.$$

During these calculations, the value of n is found, and this ultimately determines what the solution for the segment will be. The change in displacement, Δx , is found by finding $(x_1 - x_0)$, corresponding to the appropriate value of n . Since equal time intervals are used, Δt_p is used instead of t in equations (5.2.3) - (5.2.8), where $\Delta t_p = t_1 - t_0$ (based on the functions being dependent upon $t - t_0$ instead of t , due to the piecewise linearization).

C. Results

Equations (5.2.3) - (5.2.14) were assembled into a computer program, as well as a Runge-Kutta solution using DVERK [5], and the results obtained by running it on the Amdahl 470 V/8 [26].

Initially, the approximation was tested by using a small value for μ as a check that it worked. This was arbitrarily chosen to be 0.25 and was tested for x_0 being equal to 1.0, 2.0, and 2.5 with Δt_p for the piecewise linearization being equal to 0.2, and the Δt_{RK} for the Runge-Kutta solution [5] being 0.05. The size of the time intervals was chosen arbitrarily. The results are seen on Figures 5.3 - 5.8. (All solutions and phase trajectories in this chapter are based on [38],

FIGURE 5.3 - SOLUTIONS TO VAN DER POL EQUATION FOR
 $\mu = 0.25$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 1.0$,
 $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [38].)

$\mu = 0.2500$
 Δt (PIECEWISE LINEARIZATION) = 0.2000
 Δt (RUNGE-KUTTA) = 0.0500
 $x(0) = 1.0000 \quad \dot{x}(0) = 0.0000$
— RUNGE-KUTTA
○ PIECEWISE LINEARIZATION

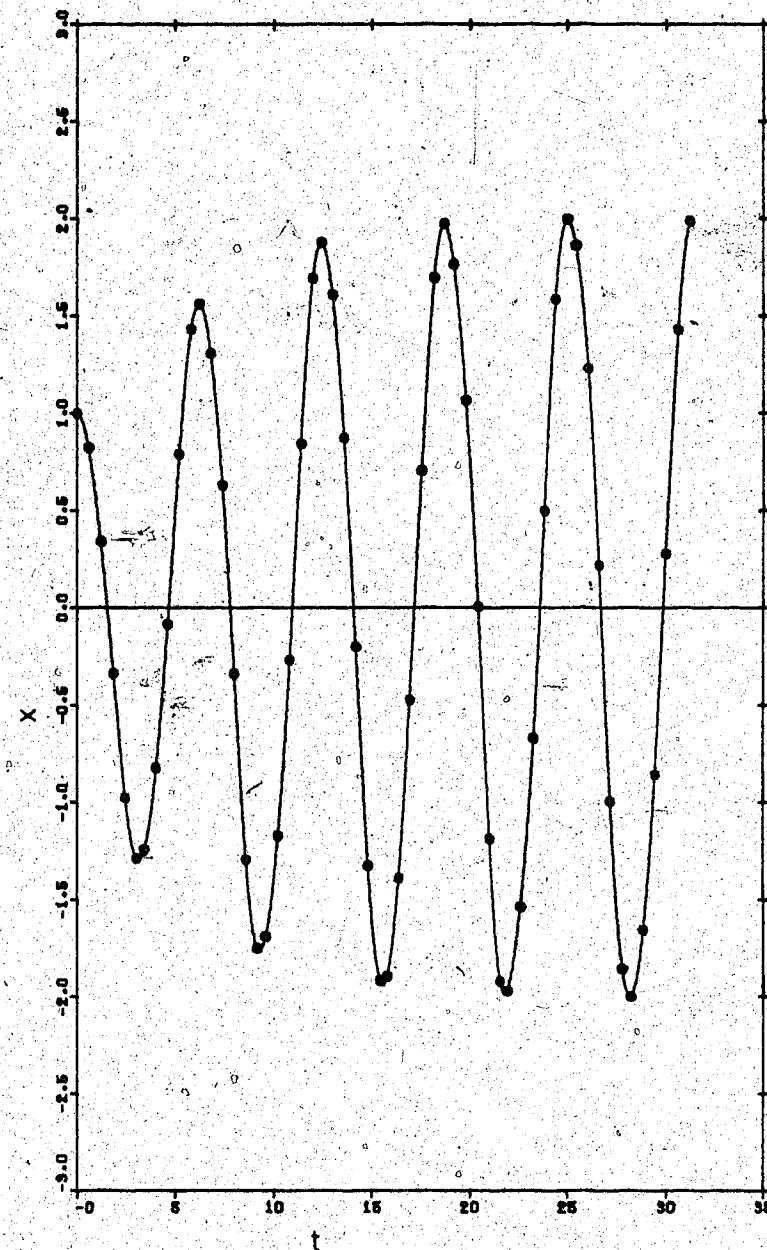


FIGURE 5.4 - PHASE TRAJECTORIES FOR VAN DER POL EQUATION FOR $\mu = 0.25$, $\Delta t^P = 0.20$, $\Delta t^{RK} = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram based on [39].)

$\mu = 0.2500$
At (PIECEWISE LINEARIZATION) = 0.2000
At (RUNGE-KUTTA) = 0.0500
 $x(0) = 1.0000 \quad \dot{x}(0) = 0.0000$

— RUNGE-KUTTA
○ PIECEWISE LINEARIZATION

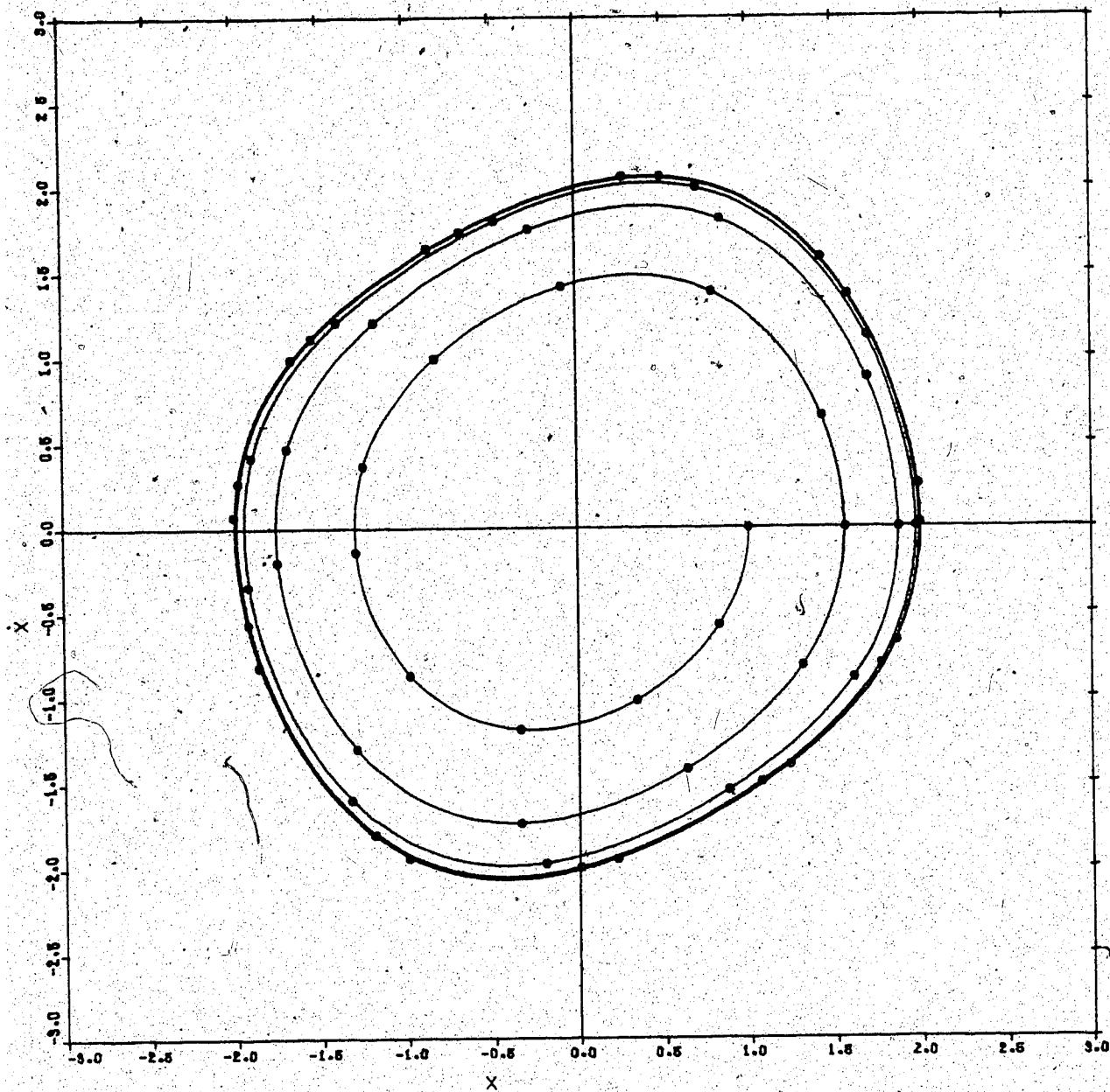


FIGURE 5.5 - SOLUTIONS TO VAN DER POL EQUATION FOR
 $\mu = 0.25$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.0$,
 $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [38].)

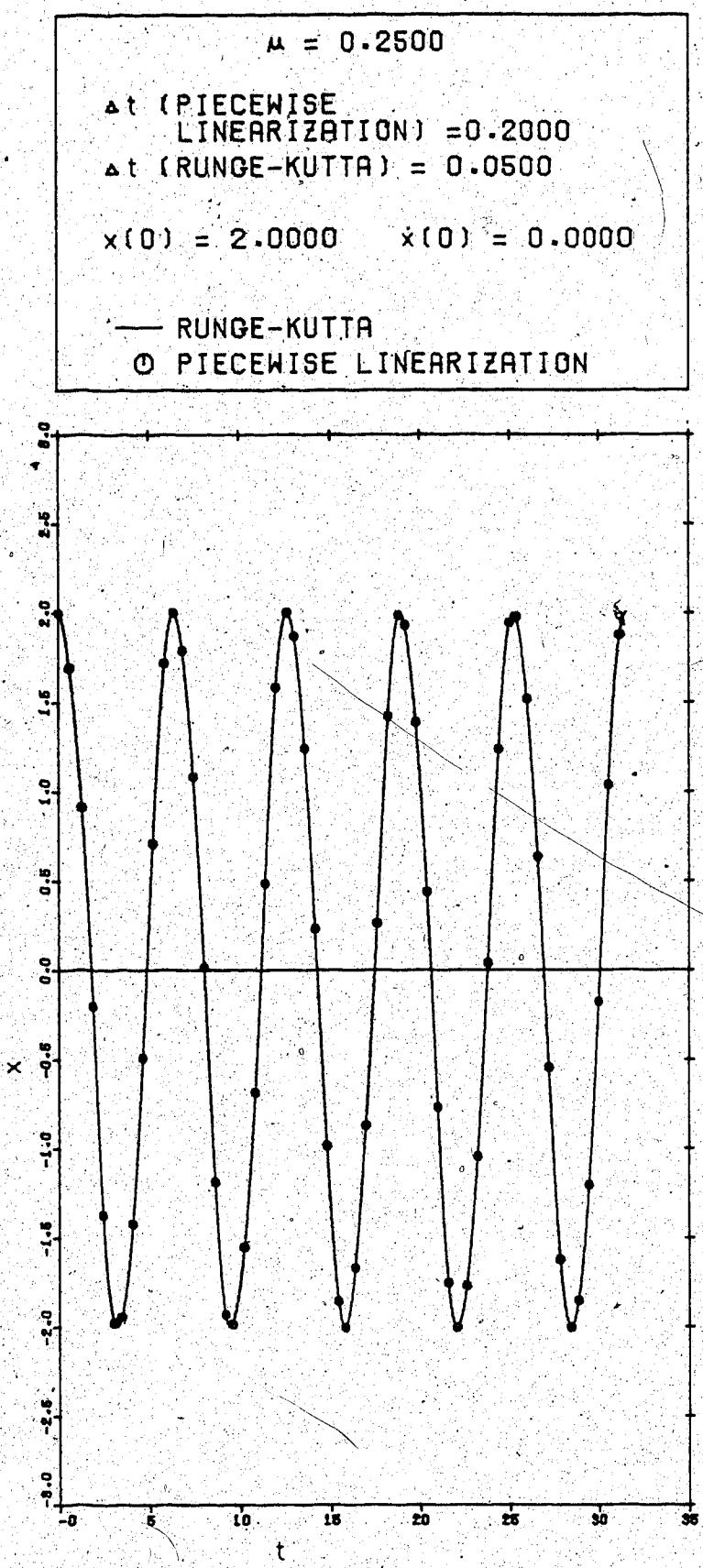




FIGURE 5.6 - PHASE TRAJECTORIES FOR VAN DER POL
EQUATION FOR $\mu = 0.25$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$,
 $x(0) = 2.0$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [39].)

$\mu = 0.2500$
 Δt (PIECEWISE LINEARIZATION) = 0.2000
 Δt (RUNGE-KUTTA) = 0.0500
 $x(0) = 2.0000 \quad \dot{x}(0) = 0.0000$

— RUNGE-KUTTA
○ PIECEWISE LINEARIZATION

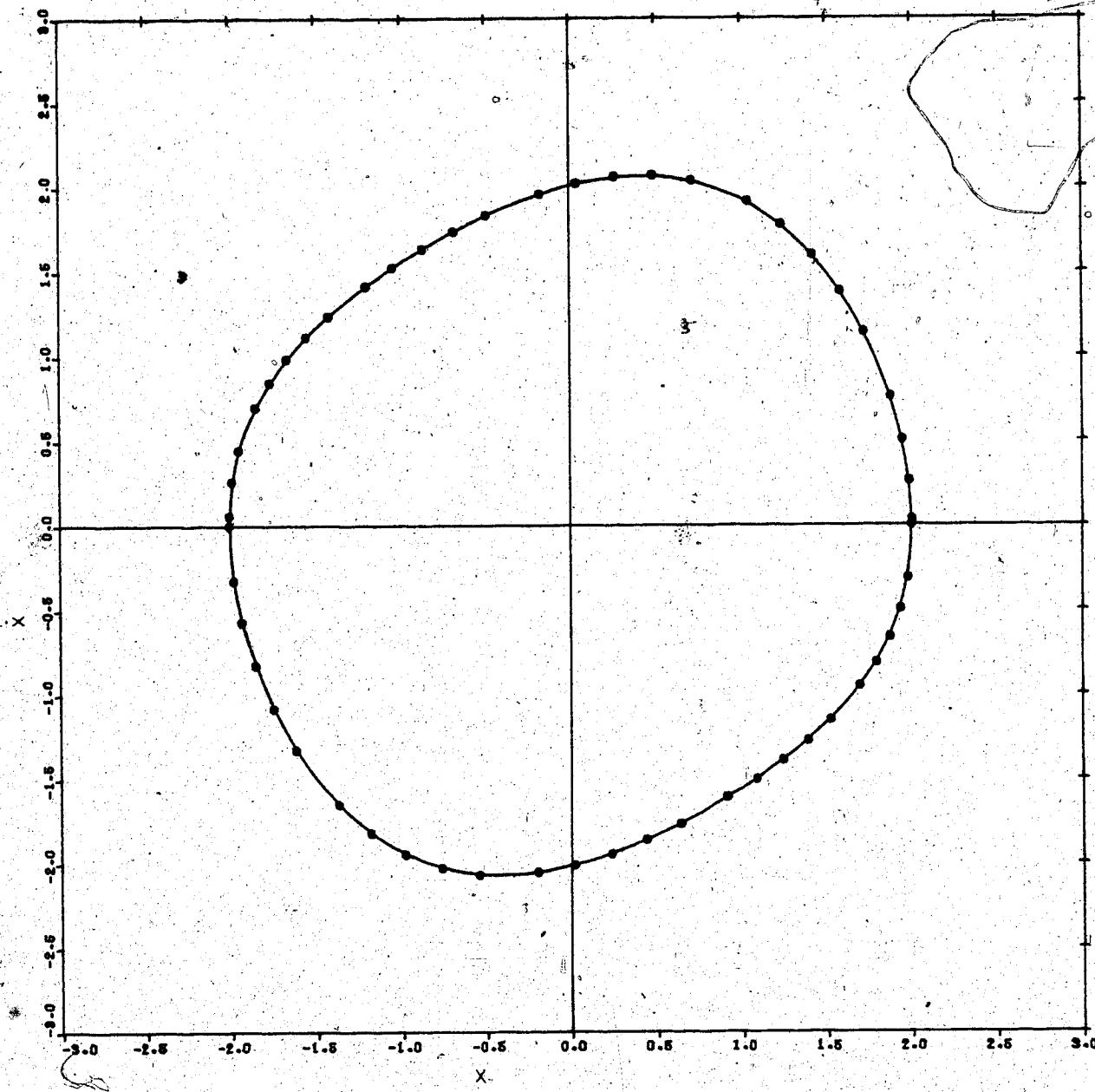


FIGURE 5.7 - SOLUTIONS TO VAN DER POL EQUATION FOR
 $\mu = 0.25$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.5$,
 $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [38].)

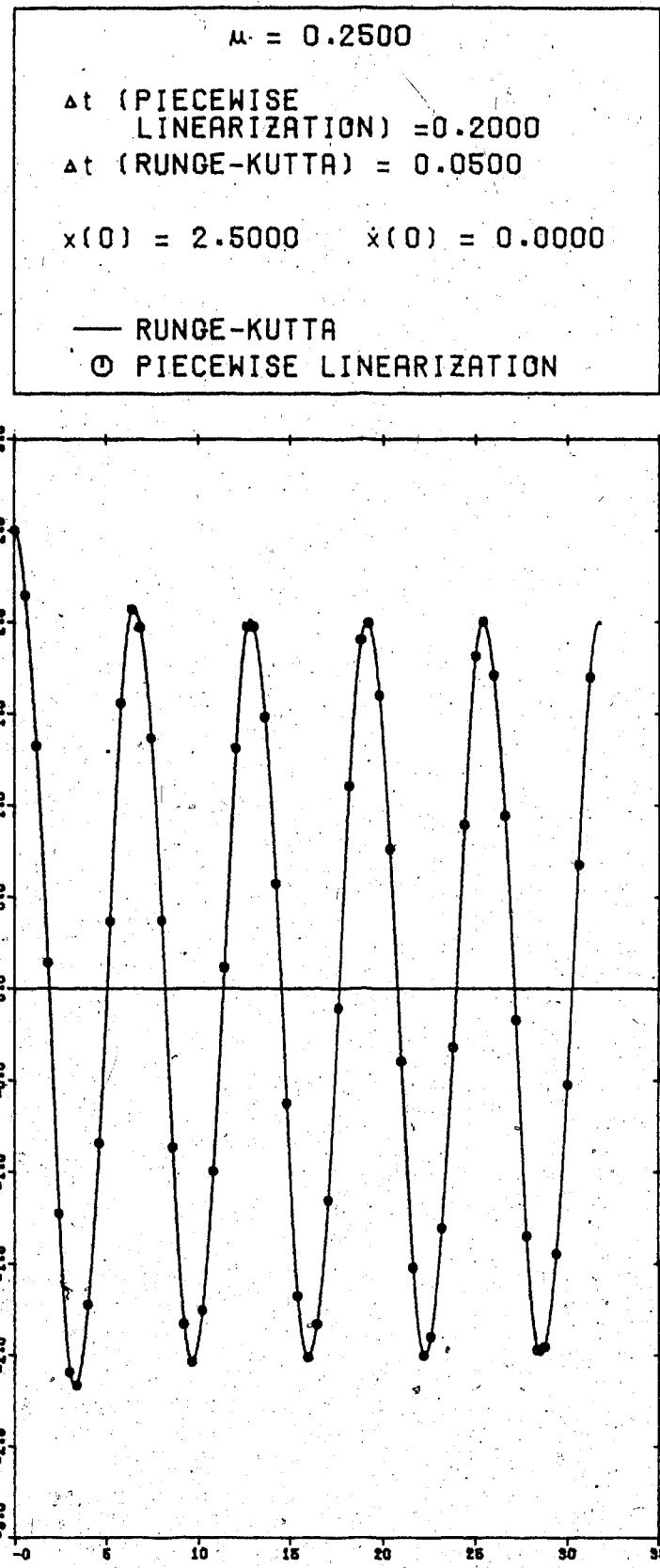


FIGURE 5.8 - PHASE TRAJECTORIES FOR VAN DER POL
EQUATION FOR $\mu = 0.25$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$,
 $x(0) = 2.5$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [39].)

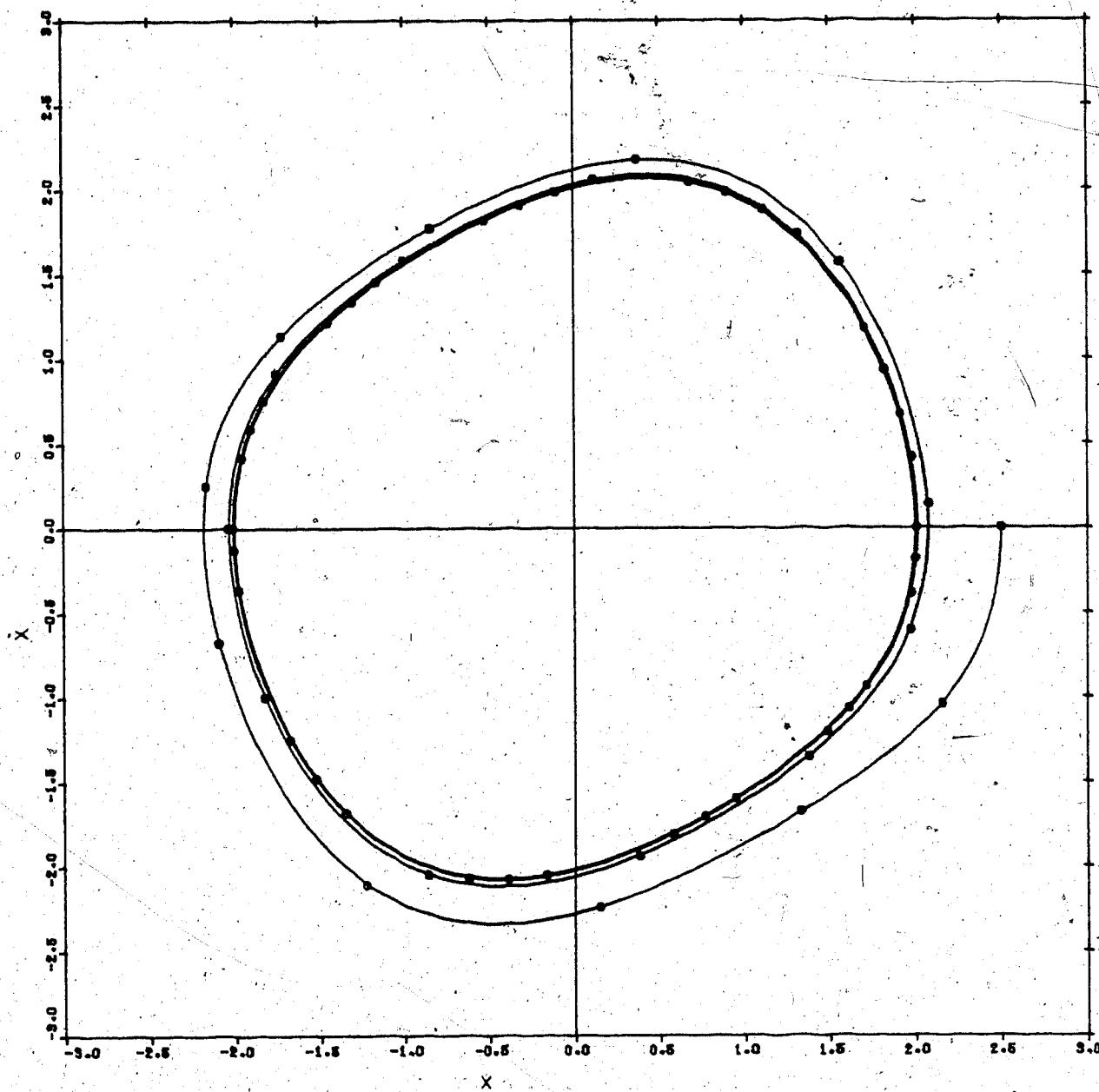
$\mu = 0.2500$

Δt (PIECEWISE LINEARIZATION) = 0.2000

Δt (RUNGE-KUTTA) = 0.0500

$x(0) = 2.5000 \quad \dot{x}(0) = 0.0000$

— RUNGE-KUTTA
○ PIECEWISE LINEARIZATION



[39].) The development of the limit cycle can be seen in the phase-plane diagrams and there appear to be no major difficulties in achieving the desired results. As expected for an initial displacement outside the limit cycle, the solution moves in towards it, while moving out to it when x_0 is inside it. Even for the initial displacement being on the limit cycle, there seem to be no problems.

The next thing was to increase the value of μ while keeping the values of the respective time intervals the same. It had also been decided to restrict the remaining investigation to the situation of the initial displacement outside the limit cycle, except for μ of 5.0 where x_0 will be varied as had been done for μ of 0.25.

The results of this can be seen in Figures 5.9 - 5.18 [38], [39], but something unusual appears in Figures 5.13 - 5.18 [38], [39]. Up to this point, the plots had given good fits to the solution being compared (with some minor deviations for μ of 2.0), while for these curves, this does not seem to be the case. Immediately the piecewise linearization was suspected to be at fault and both the derivation and the computer program were checked several times. Nothing that could introduce what would appear to be a phase difference was found. Internal roundoff error in the computer was also suspected, and the tolerances were reset to smaller values and, as far as possible, the program was reset to operate in double precision. But, the results were still the same--a phase difference, or what appeared to be as such, somehow arose in the system.

A smaller time interval of 0.05 was tried for the piecewise linearization, with the results for $x(0) = 2.5$ and $\mu = 5.0$ to be seen in Figures 5.19 and 5.20 [38], [39]. This seems to be the key to the

FIGURE 5.9 - SOLUTIONS TO VAN DER POL EQUATION FOR
 $\mu = 1.0$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.5$,
 $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [38].)

$\mu = 1.0000$
 Δt (PIECEWISE LINEARIZATION) = 0.2000
 Δt (RUNGE-KUTTA) = 0.0600
 $x(0) = 2.5000 \quad x(0) = 0.0000$

— RUNGE-KUTTA
○ PIECEWISE LINEARIZATION

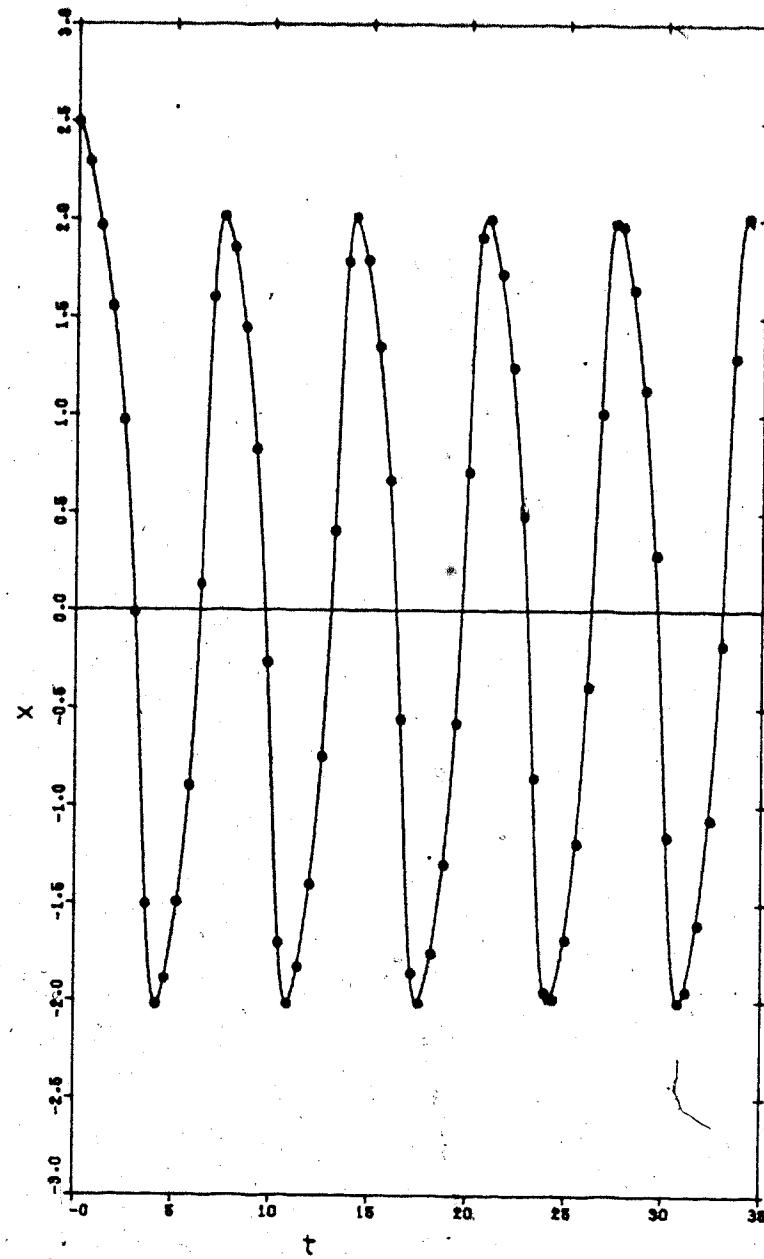


FIGURE 5.10 - PHASE TRAJECTORIES FOR VAN DER POL
EQUATION FOR $\mu = 1.0$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$,
 $x(0) = 2.5$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [39].)

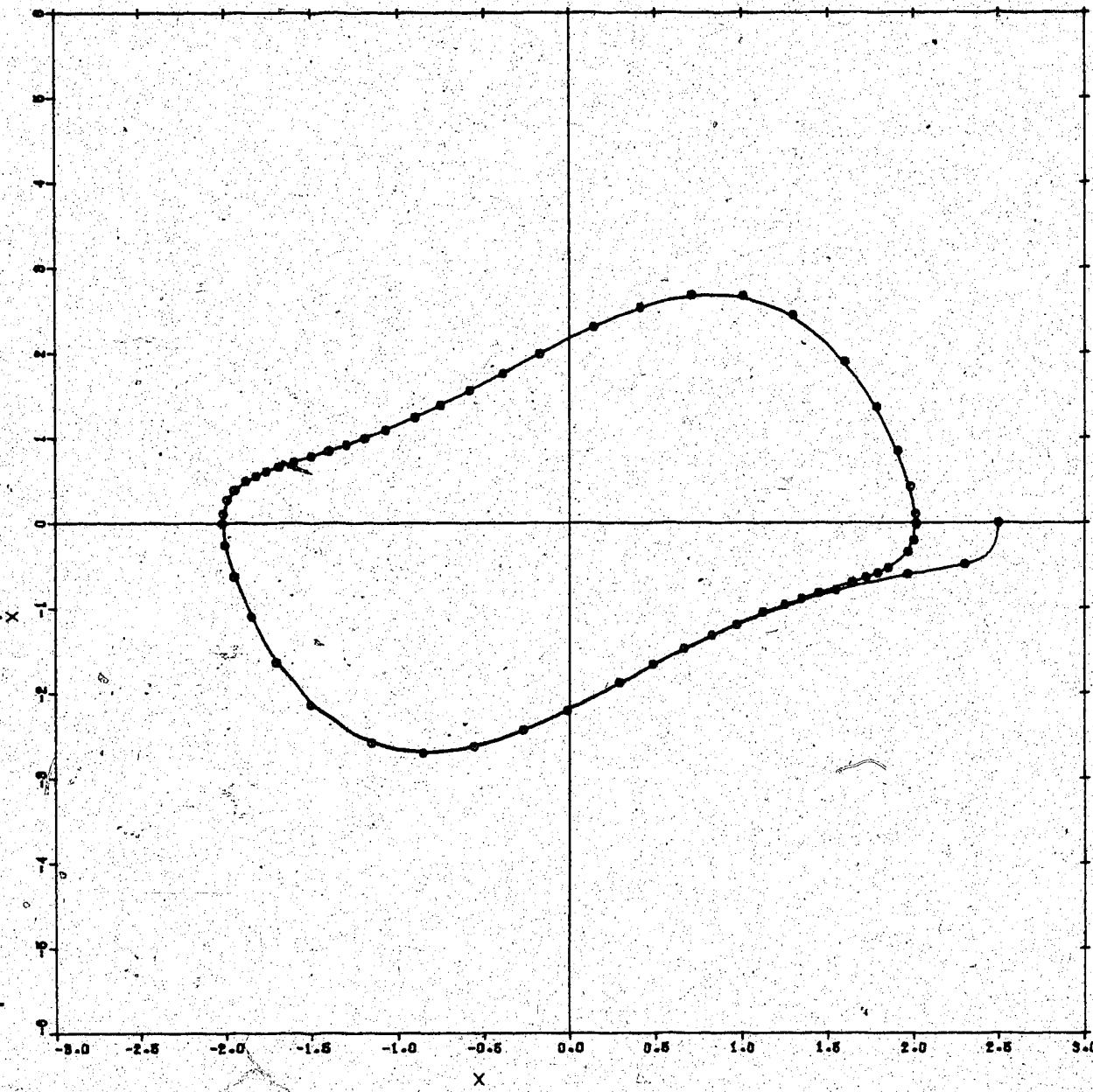
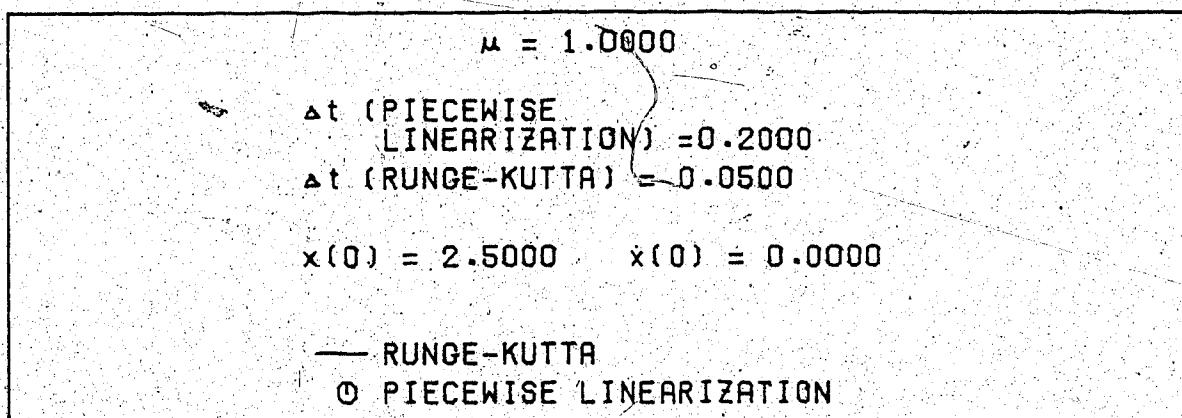


FIGURE 5.11 - SOLUTIONS TO VAN DER POL EQUATION FOR
 $\mu = 2.0$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.5$,
 $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [38].)

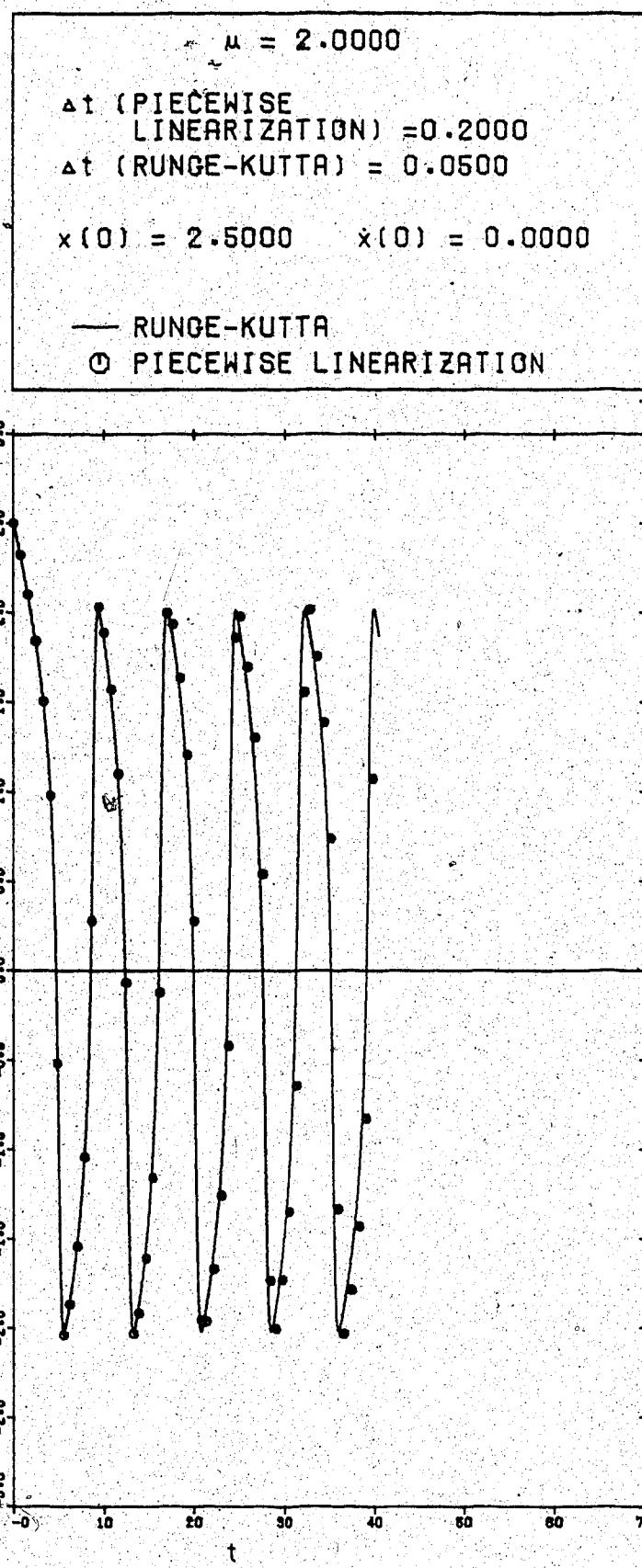


FIGURE 5.12 - PHASE TRAJECTORIES FOR VAN DER POL
EQUATION FOR $\mu = 2.0$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$,
 $x(0) = 2.5$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [39].)

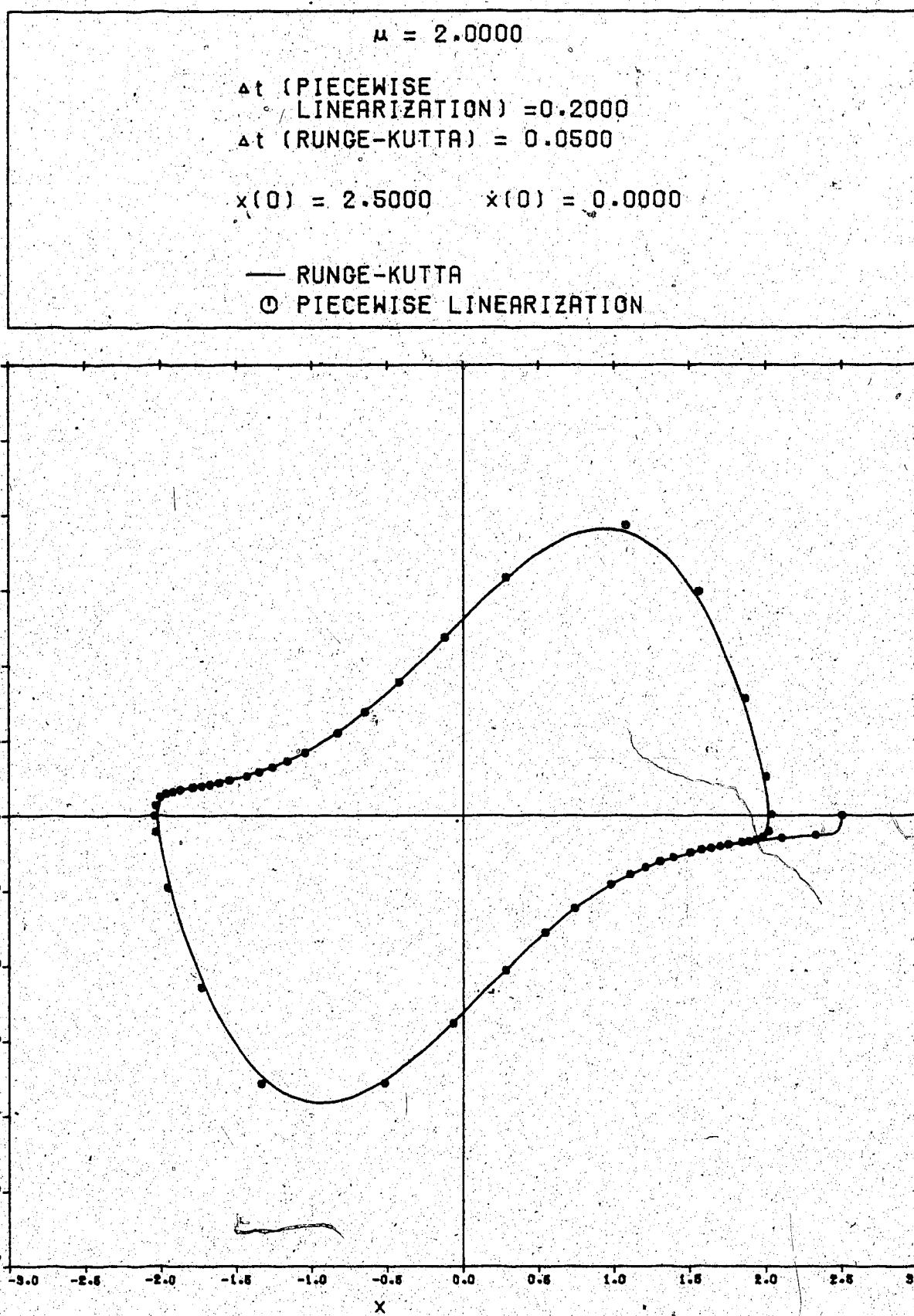


FIGURE 5.13 - SOLUTIONS TO VAN DER POL EQUATION FOR
 $\mu = 5.0$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 1.0$,
 $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [38].)

$\mu = 5.0000$
 Δt (PIECEWISE LINEARIZATION) = 0.2000
 Δt (RUNGE-KUTTA) = 0.0500
 $x(0) = 1.0000 \quad \dot{x}(0) = 0.0000$
— RUNGE-KUTTA
○ PIECEWISE LINEARIZATION

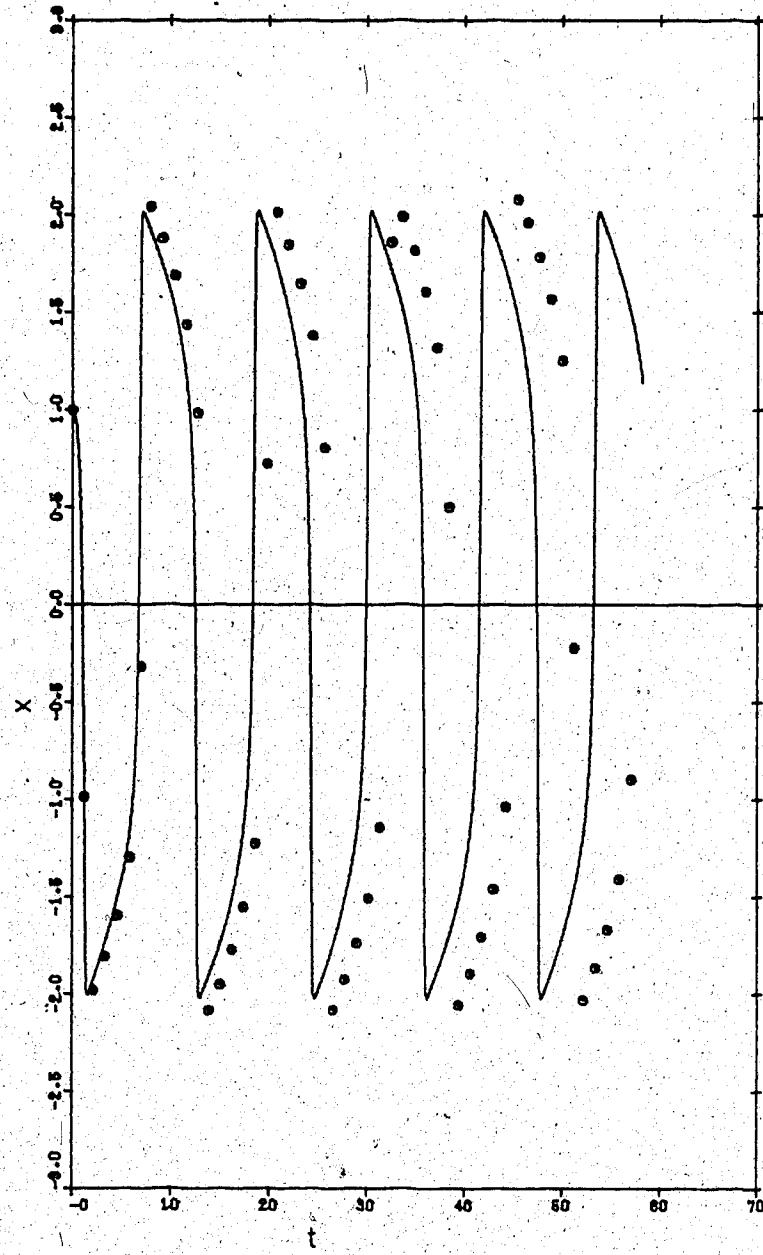


FIGURE 5.14 - PHASE TRAJECTORIES FOR VAN DER POL
EQUATION FOR $\mu = 5.0$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$,
 $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [39].)

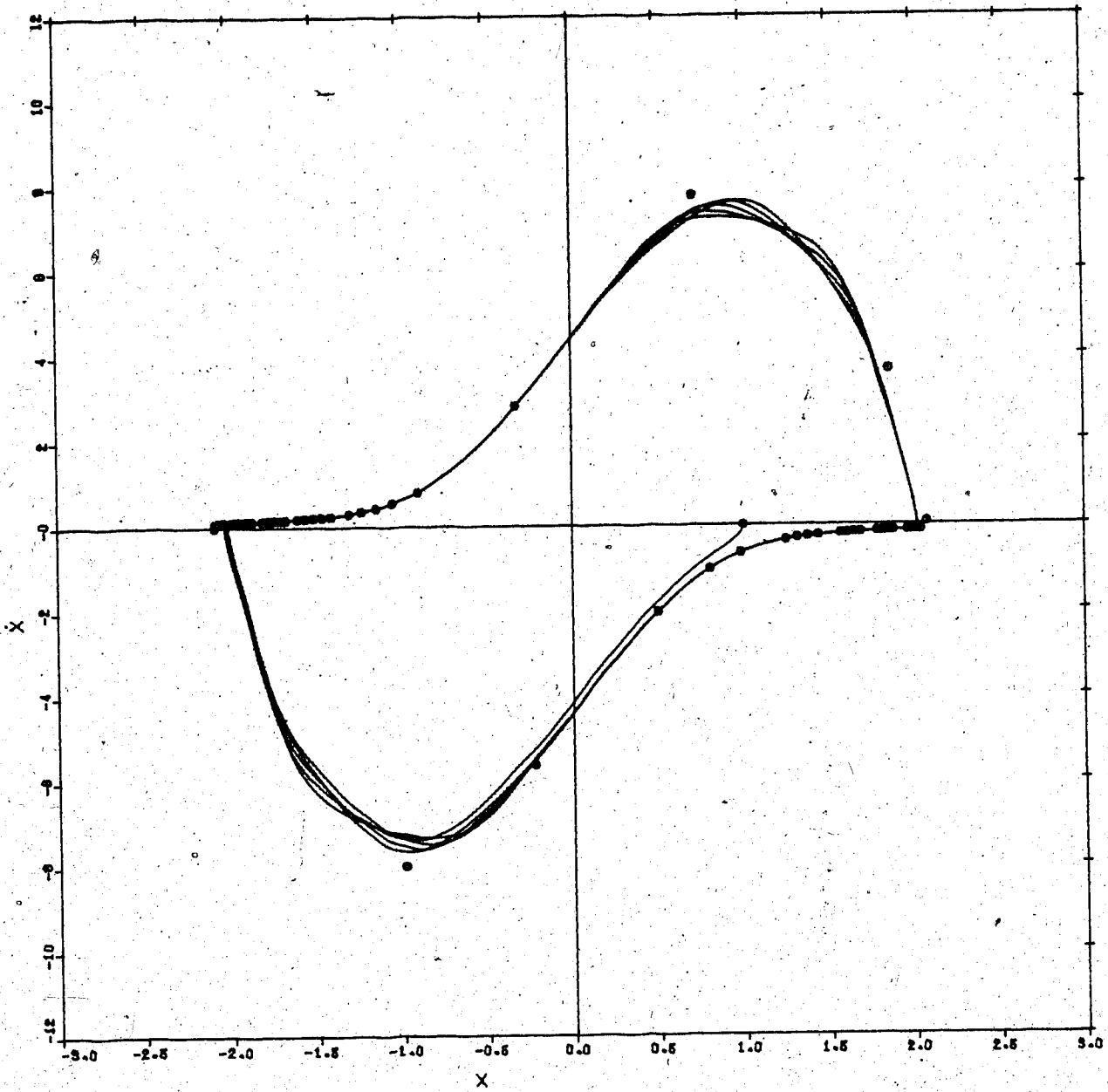
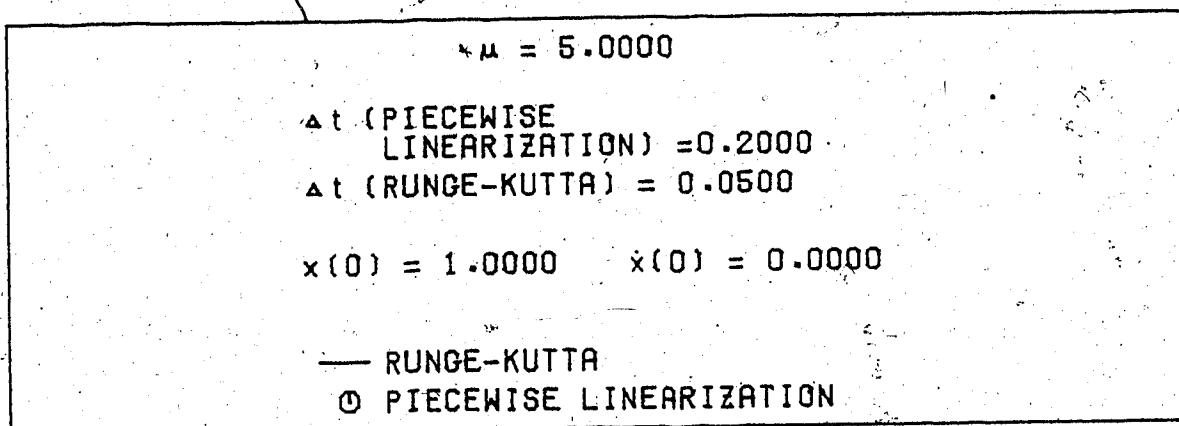


FIGURE 5.15 - SOLUTIONS TO VAN DER POL EQUATION FOR
 $\mu = 5.0$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.0$,
 $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [38].)

$\mu = 5.0000$

At (PIECEWISE LINEARIZATION) = 0.2000
At (RUNGE-KUTTA) = 0.0500

$x(0) = 2.0000 \quad \dot{x}(0) = 0.0000$

— RUNGE-KUTTA
○ PIECEWISE LINEARIZATION

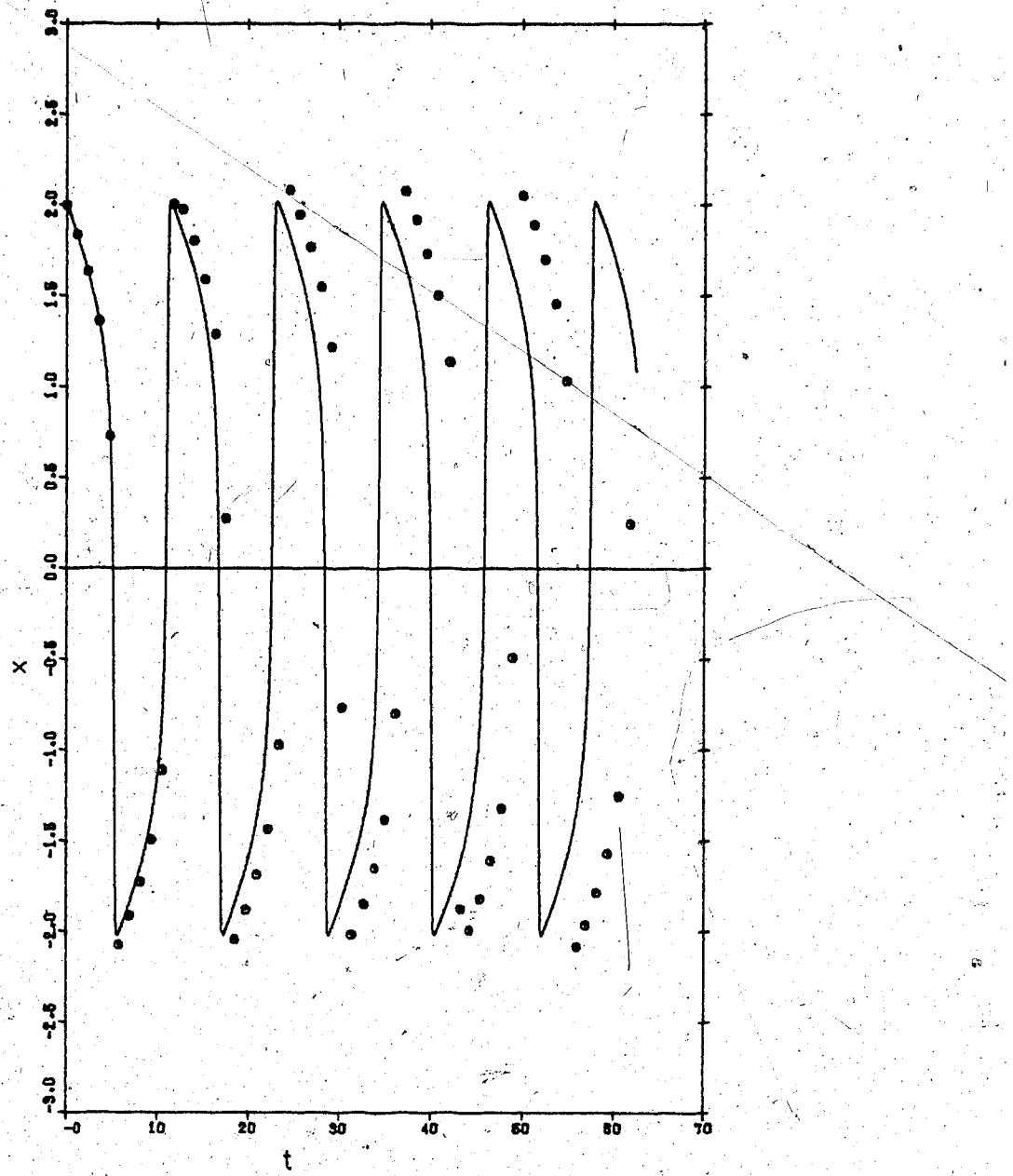


FIGURE 5.16 - PHASE TRAJECTORIES FOR VAN DER POL
EQUATION FOR $\mu = 5.0$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$,
 $x(0) = 2.0$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [39].)

$\mu = 6.0000$
 Δt (PIECEWISE LINEARIZATION) = 0.2000
 Δt (RUNGE-KUTTA) = 0.0500
 $x(0) = 2.0000$ $\dot{x}(0) = 0.0000$
— RUNGE-KUTTA
○ PIECEWISE LINEARIZATION

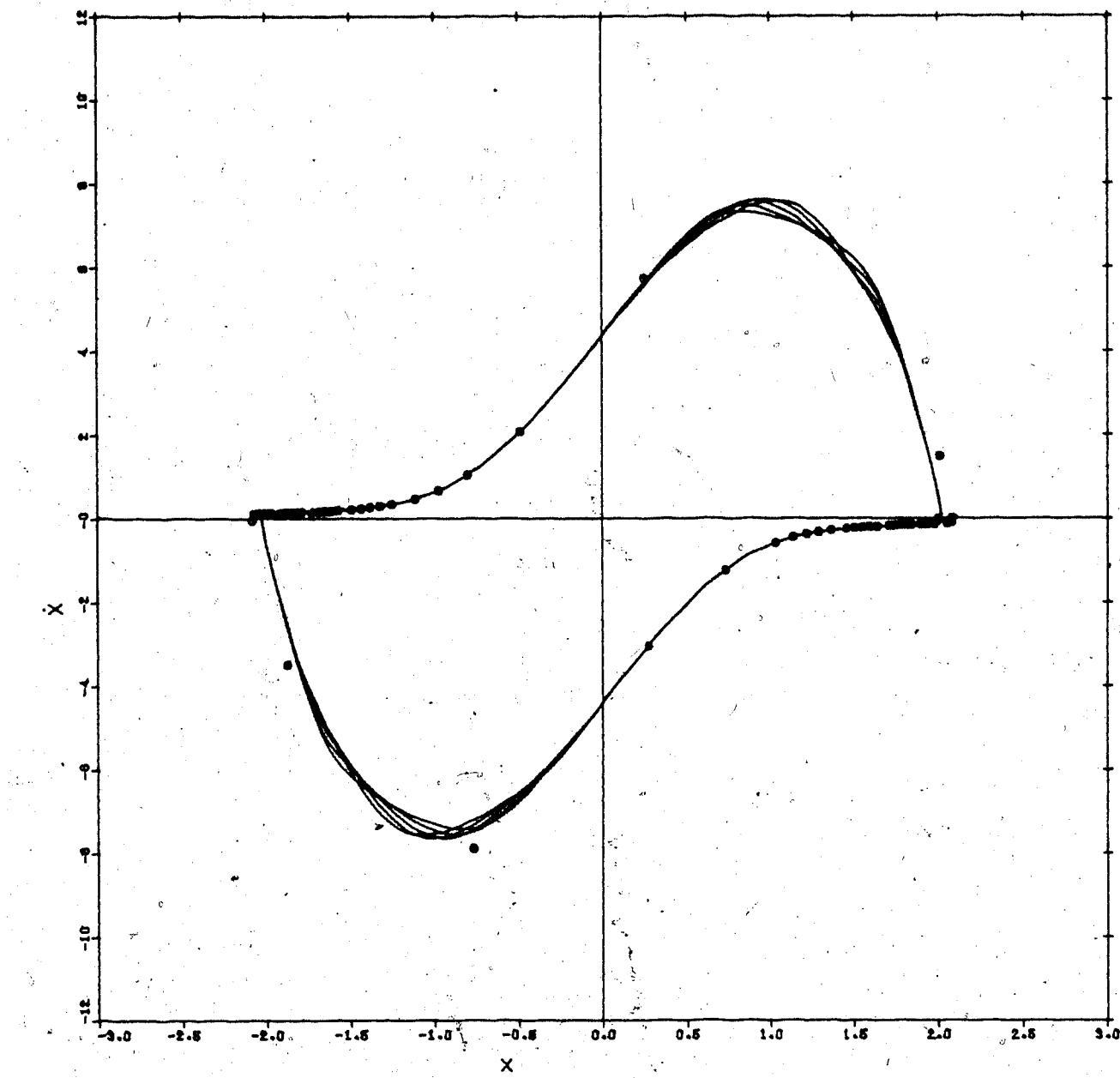


FIGURE 5.17 - SOLUTIONS TO VAN DER POL EQUATION FOR
 $\mu = 5.0$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.5$,
 $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [38].)

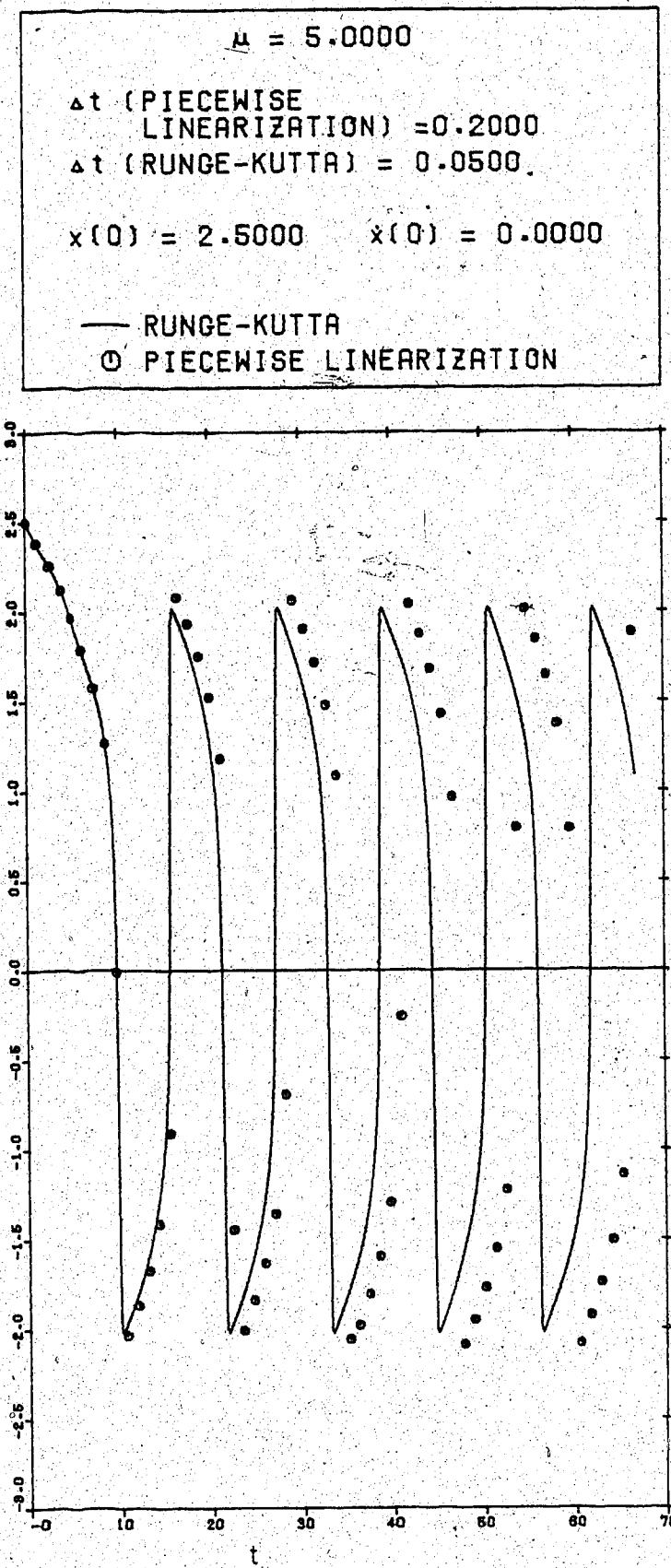


FIGURE 5.18 - PHASE TRAJECTORIES FOR VAN DER POL
EQUATION FOR $\mu = 5.0$, $\Delta t_p = 0.20$, $\Delta t_{RK} = 0.05$,
 $x(0) = 2.5$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [39].)

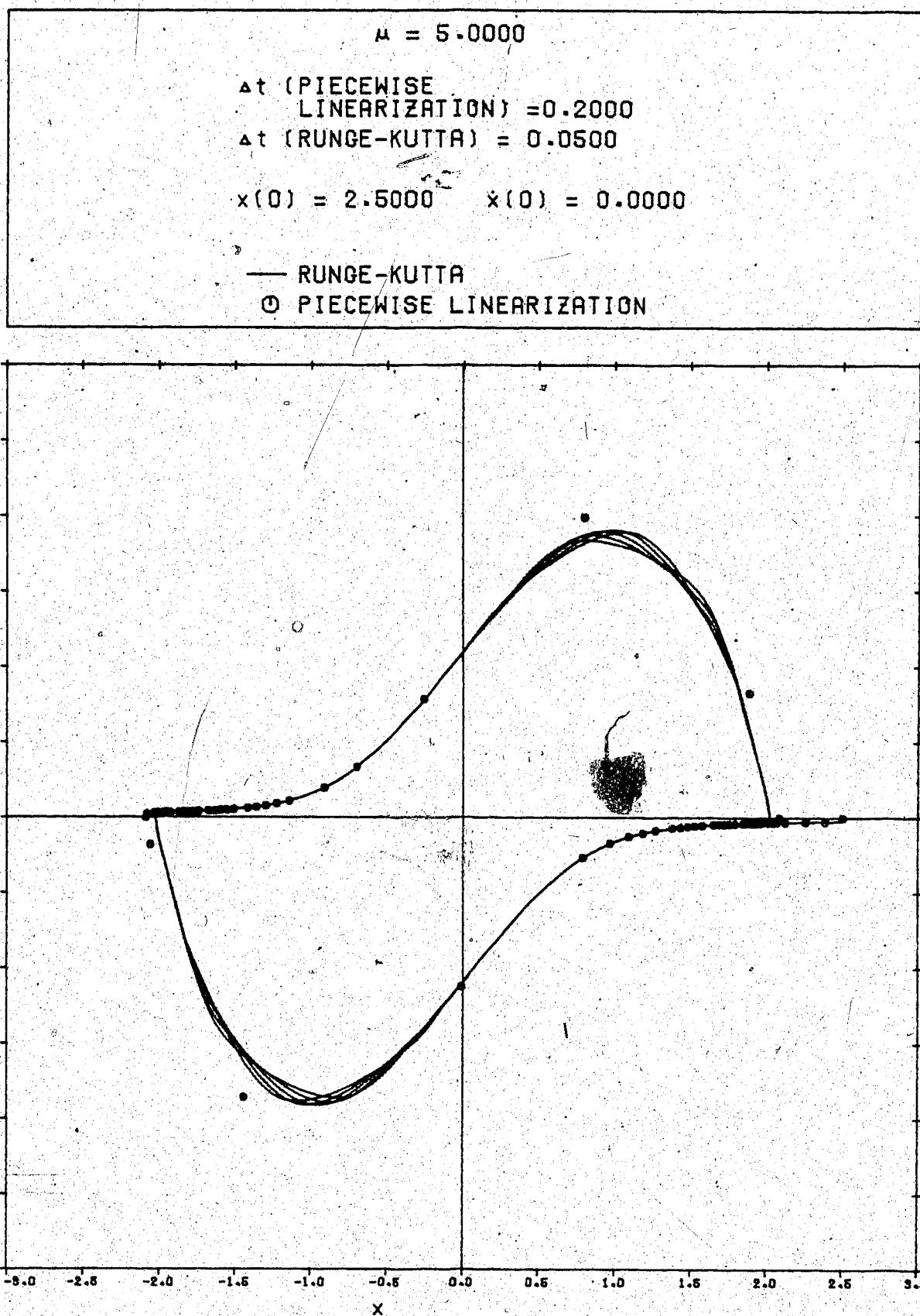


FIGURE 5.19 - SOLUTIONS TO VAN DER POL EQUATION FOR
 $\mu = 5.0$, $\Delta t_p = 0.05$, $\Delta t_{RK} = 0.05$, $x(0) = 2.5$,
 $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [38].)

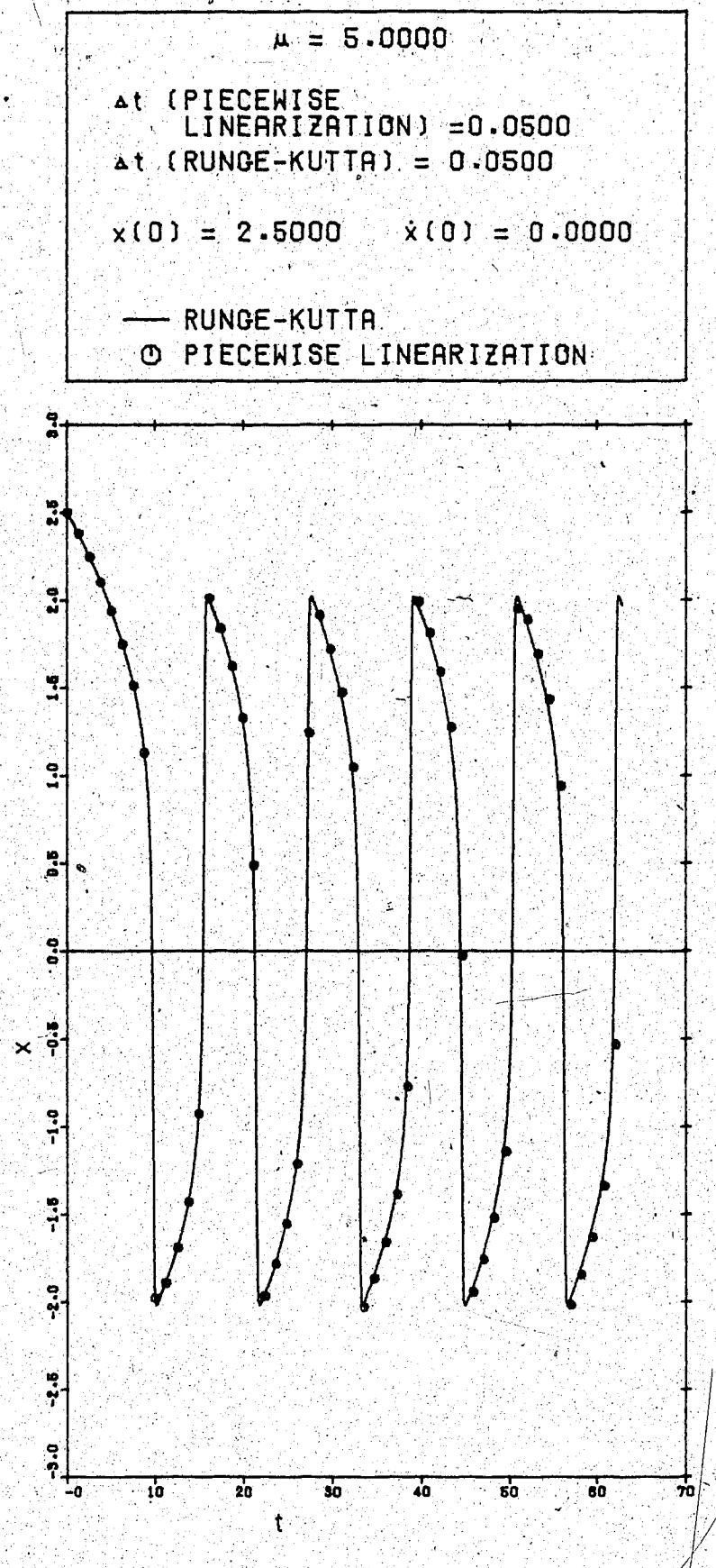


FIGURE 5.20.- PHASE TRAJECTORIES FOR VAN DER POL
EQUATION FOR $\mu = 5.0$, $\Delta t_p = 0.05$, $\Delta t_{RK} = 0.05$,
 $x(0) = 2.5$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [39].)

$\mu = 5.0000$
at (PIECEWISE LINEARIZATION) = 0.0500
at (RUNGE-KUTTA) = 0.0500
 $x(0) = 2.5000 \quad x(0) = 0.0000$

— RUNGE-KUTTA
○ PIECEWISE LINEARIZATION

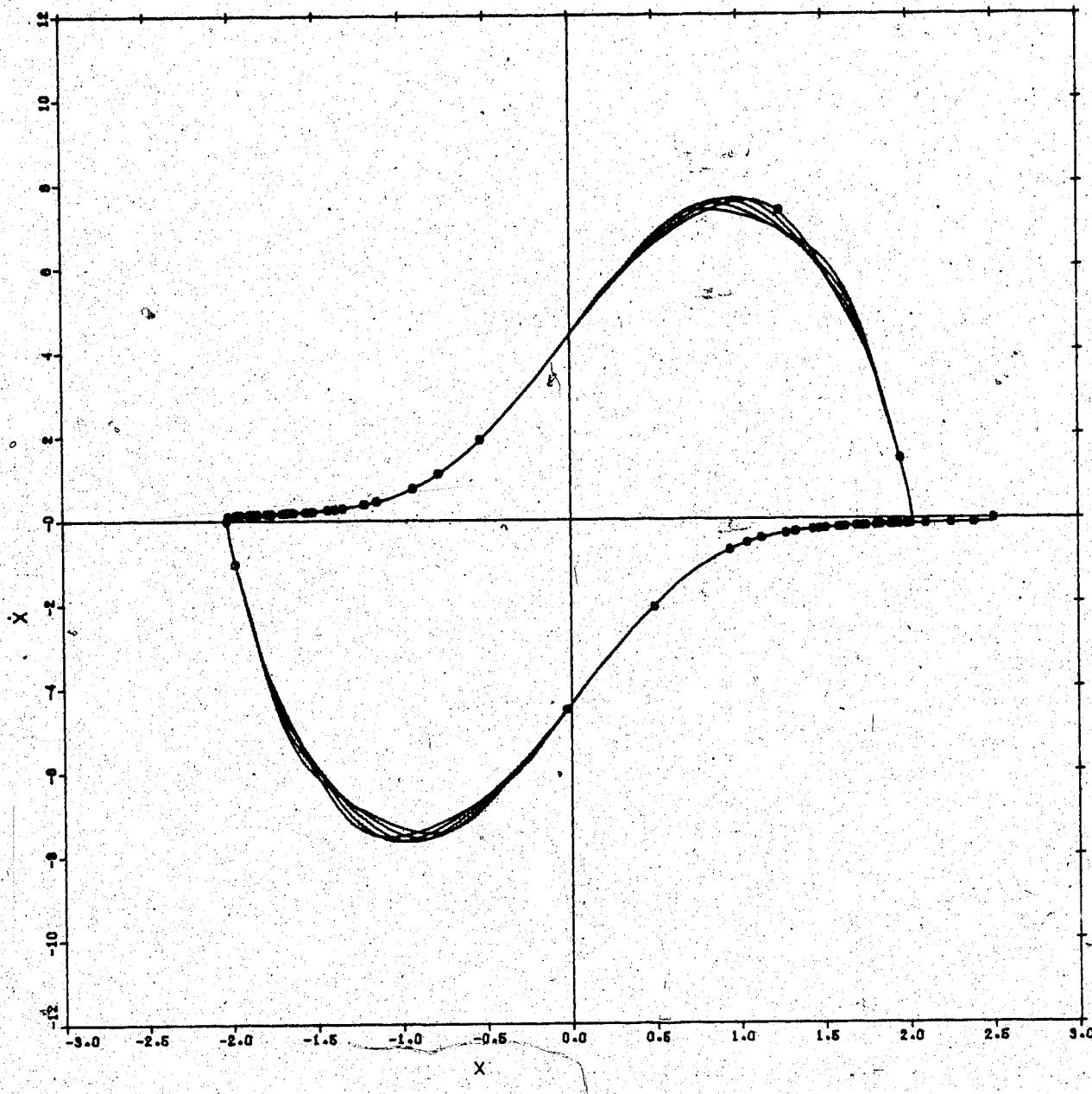


FIGURE 5.21 - SOLUTIONS TO VAN DER POL EQUATION FOR
 $\mu = 5.0$, $\Delta t_p = 0.25$, $\Delta t_{RK} = 0.25$, $x(0) = 2.5$,
 $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [38].)



$\mu = 5.0000$
 Δt (PIECEWISE LINEARIZATION) = 0.2500
 Δt (RUNGE-KUTTA) = 0.2500
 $x(0) = 2.5000 \quad \dot{x}(0) = 0.0000$

— RUNGE-KUTTA
○ PIECEWISE LINEARIZATION

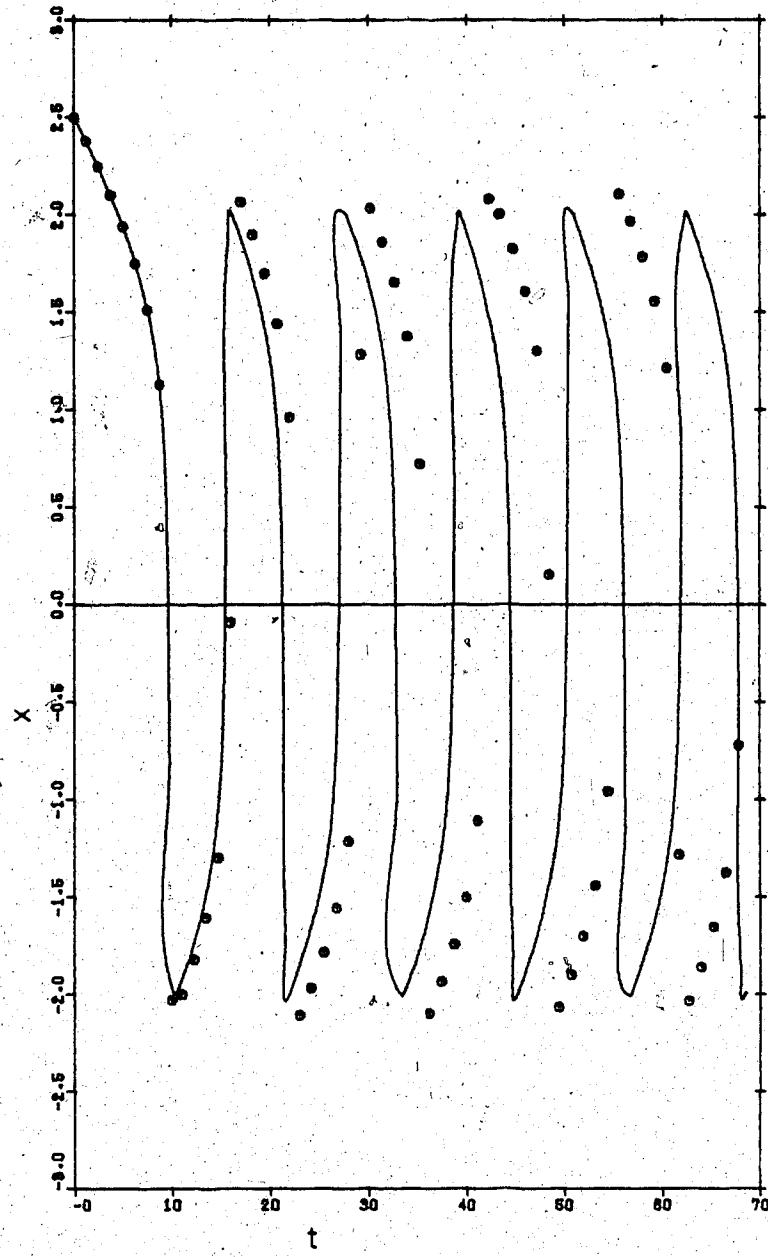


FIGURE 5.22 - PHASE TRAJECTORIES FOR VAN DER POL
EQUATION FOR $\mu = 5.0$, $\Delta t_p = 0.25$, $\Delta t_{RK} = 0.25$,
 $x(0) = 2.5$, $\dot{x}(0) = 0.0$

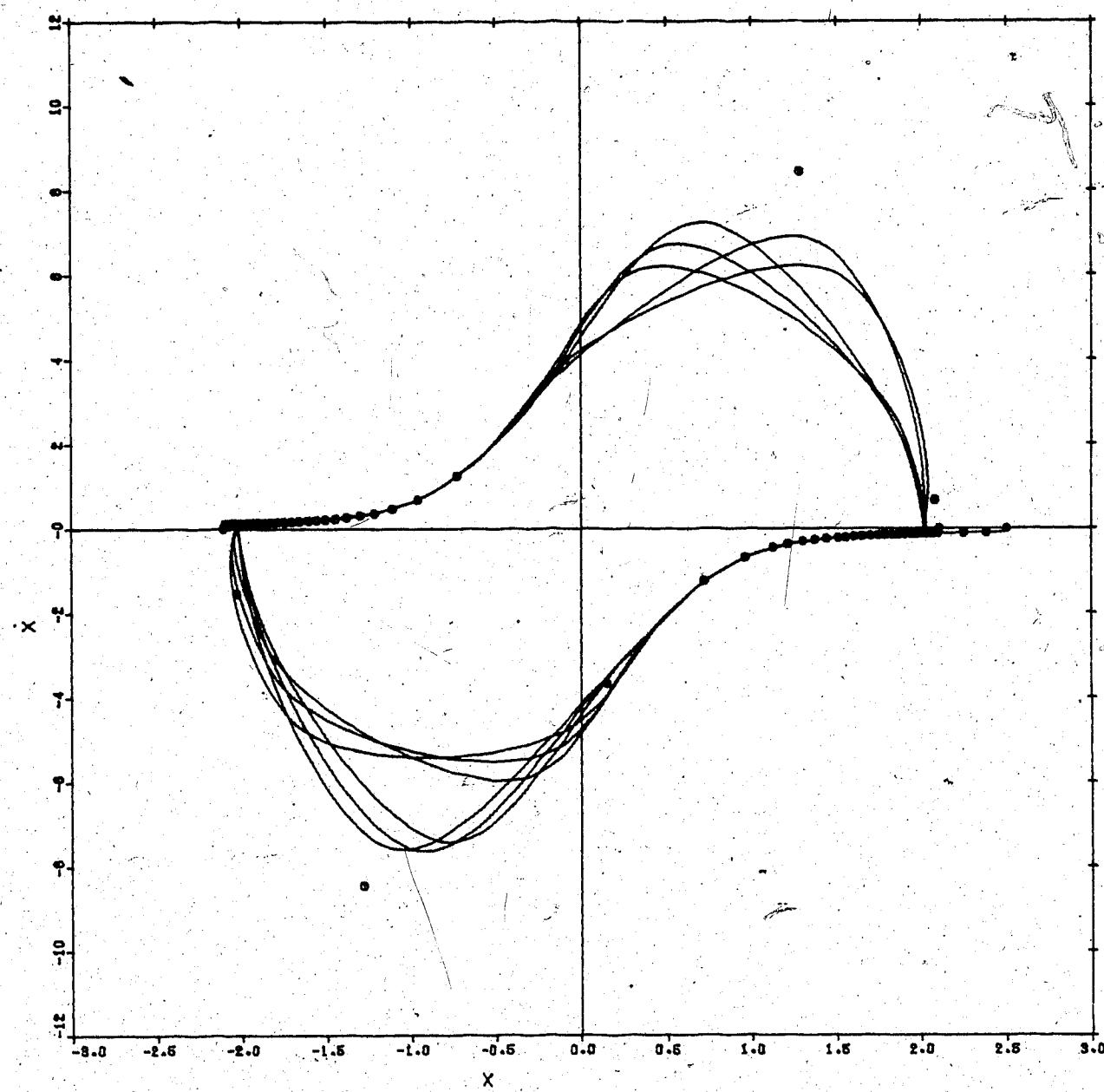
(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [39].)

$\mu = 5.0000$

at (PIECEWISE LINEARIZATION) = 0.2500
at (RUNGE-KUTTA) = 0.2500

$x(0) = 2.5000 \quad \dot{x}(0) = 0.0000$

— RUNGE-KUTTA
○ PIECEWISE LINEARIZATION



problem, since the situation appeared to be rectified to some extent in these diagrams, and completely resolved for $\mu = 2.0$ over the time span examined though the plots in question are not shown here.

This raises some interesting questions concerning the solution of this equation. One is concerned with stability, but this will be examined later as a similar problem occurs in the next chapter. Another is that if the piecewise linear method is affected by this behaviour, then the Runge-Kutta solution [5] as used here may, at times, also be suspect.

An example of the latter can be seen in Figures 5.21 and 5.22 [38], [39]. The time interval for each solution is rather large (keeping μ at 5.0), and some distortion can be seen in the displacement-time curve, while the phase plane diagram is very much affected by this to the extent that one cannot tell which part of the curve is the limit cycle.

Figure 5.21 [38] displays a paradox, as there would appear to be more than one value for the displacement as calculated by DVERK [5] for a given time (such as for t of about 23, where the value of x would be about 2.1, 1.6, and -0.9). This is physically impossible, and can lead to an erroneous conclusion about the results.

A possible source may be that the graphics routine used [40] was unable to produce a smooth curve due to a low number of data points being plotted since Δt was large. (By comparison, the curve for Figure 5.19 [38] was more accurate since a much larger number of values had been used.) The examination of the calculated numerical values may give further insight as to what may have brought this about.

Of interest now are the execution times for the two solutions. As before, the program was stripped of all non-essential steps, such as

comment statements and write commands and then run for various values of μ , Δt_p , and Δt_{RK} , with $x(0) = 3.0$ and $\dot{x}(0) = 0.0$ using [27]. The runs were for the time taken to complete 10 half-cycles for the piecewise linearization and the closest time to that value for the Runge-Kutta solution [5]. These were obtained by running an earlier version of the program on the Amdahl 470V/7 [25], with the results seen in Figures 5.23 - 5.26. As expected for this system, the Runge-Kutta [5] results were significantly lower than those for the piecewise linearization, but noteworthy is the fact that the times go up for the piecewise linear method, after $\Delta t_p = 0.25$ for $\mu = 2.0$ and $\mu = 5.0$. The calculations for the damping coefficient described in the derivation may explain why this arises, because of the number of iterations required to achieve a solution. Also, for these values of μ , the results were to be obtained for $\Delta t_p = 0.5$, as the program kept exceeding its set time limit, even after a setting of 15.0 seconds [31]. It was decided to abandon any further attempts as the runs would have proved too costly, but this might indicate that an economic upper limit may exist for these values of μ .

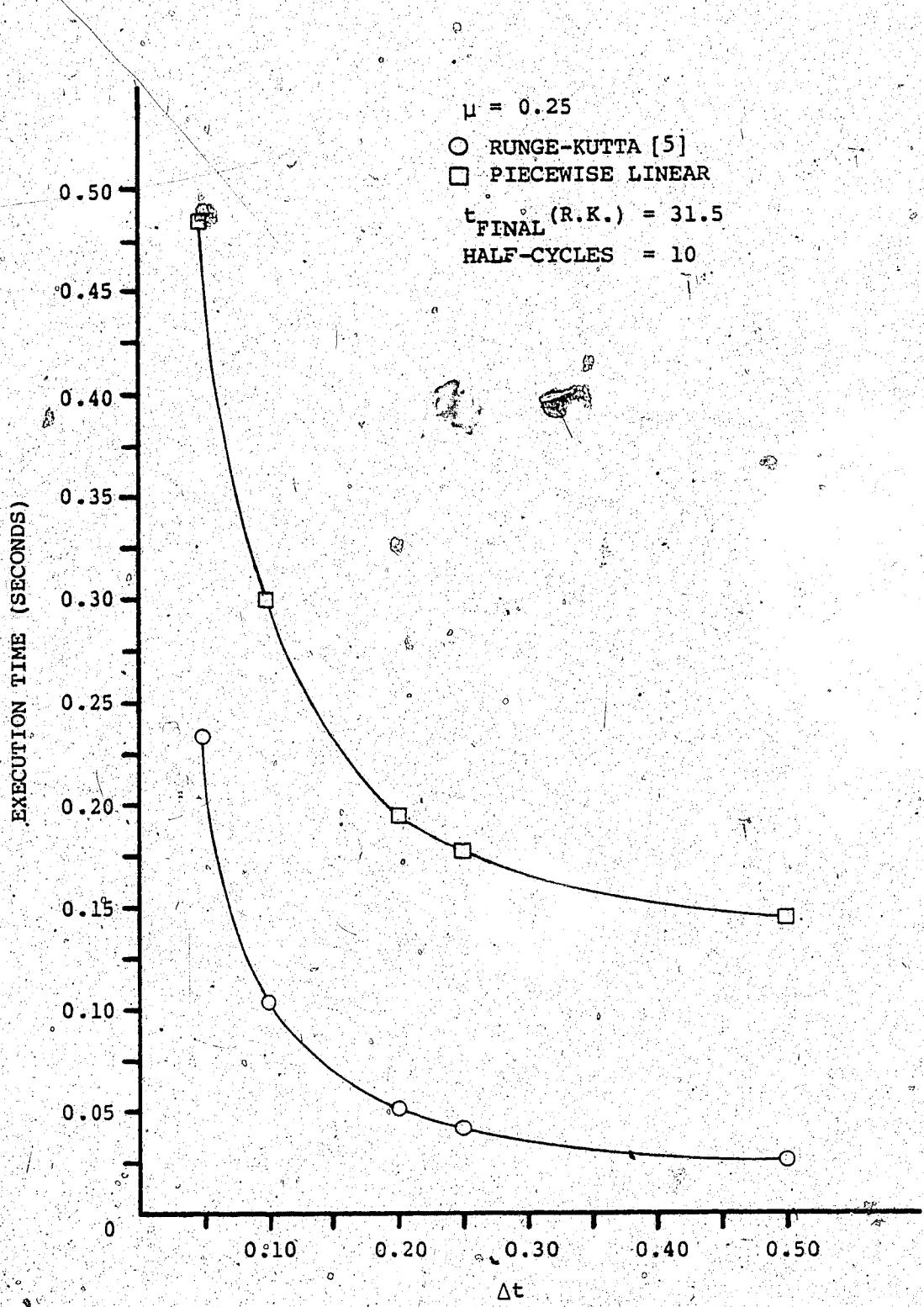


FIGURE 5.23 - EXECUTION TIMES FOR VAN DER POL EQUATION SOLUTIONS FOR $\mu = 0.25$, $x(0) = 3.0$, $\dot{x}(0) = 0.0$ [27]

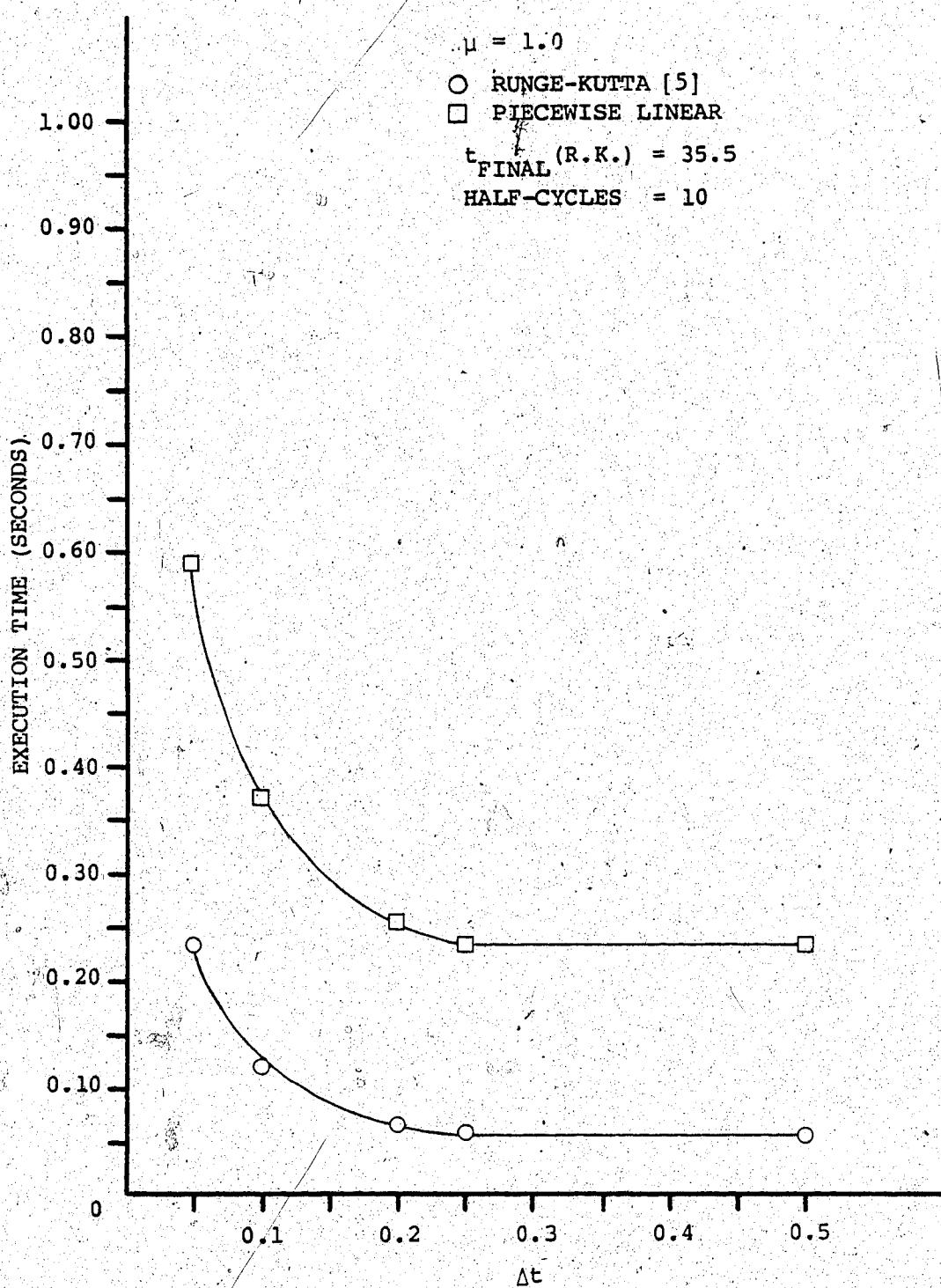


FIGURE 5.24 - EXECUTION TIMES FOR VAN DER POL EQUATION SOLUTIONS FOR $\mu = 1.0$, $x(0) = 3.0$, $\dot{x}(0) = 0.0$, [27]

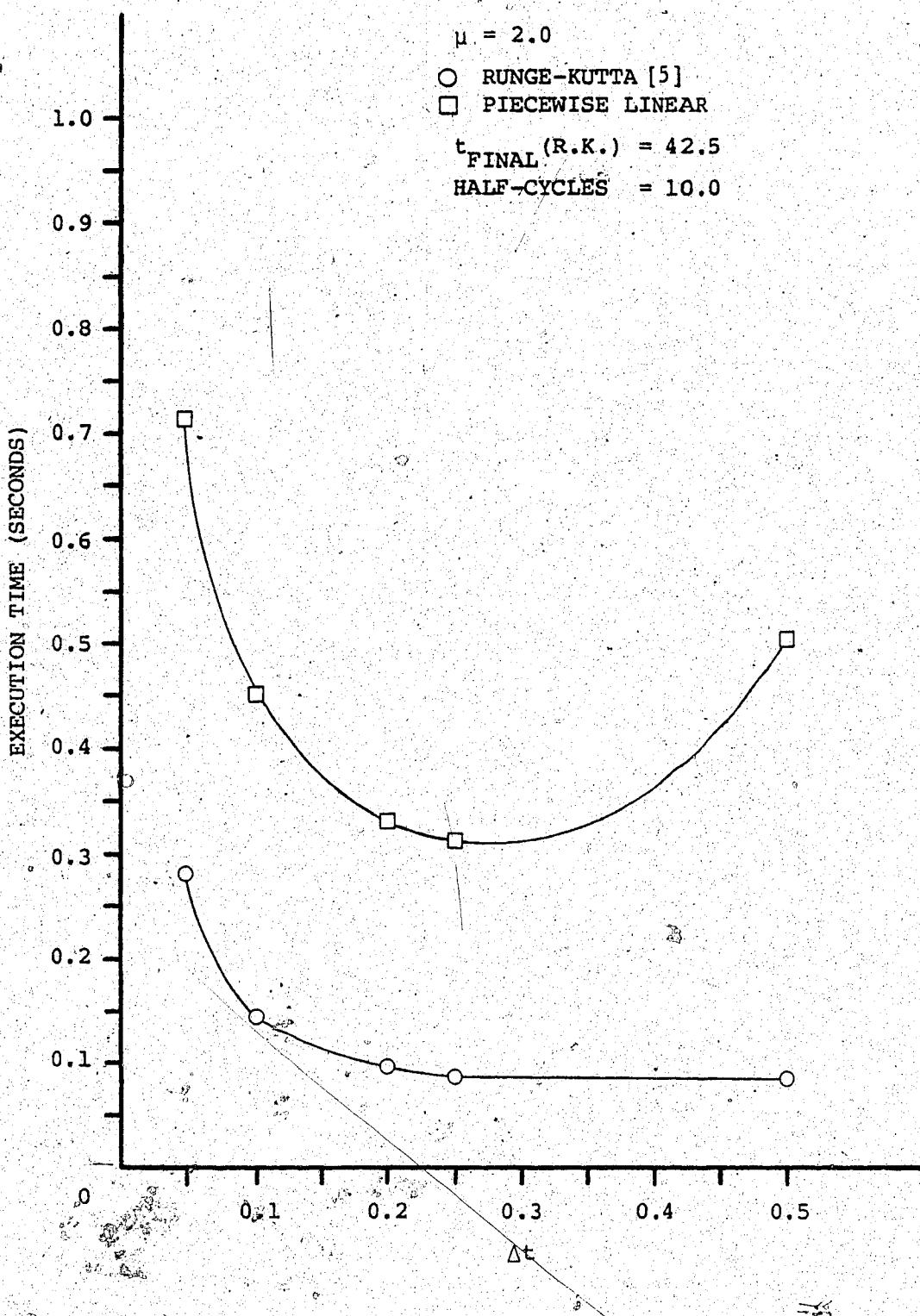


FIGURE 5.25 - EXECUTION TIMES FOR VAN DER POL EQUATION SOLUTIONS FOR $\mu = 2.0$, $x(0) = 3.0$, $\dot{x}(0) = 0.0$ [27].

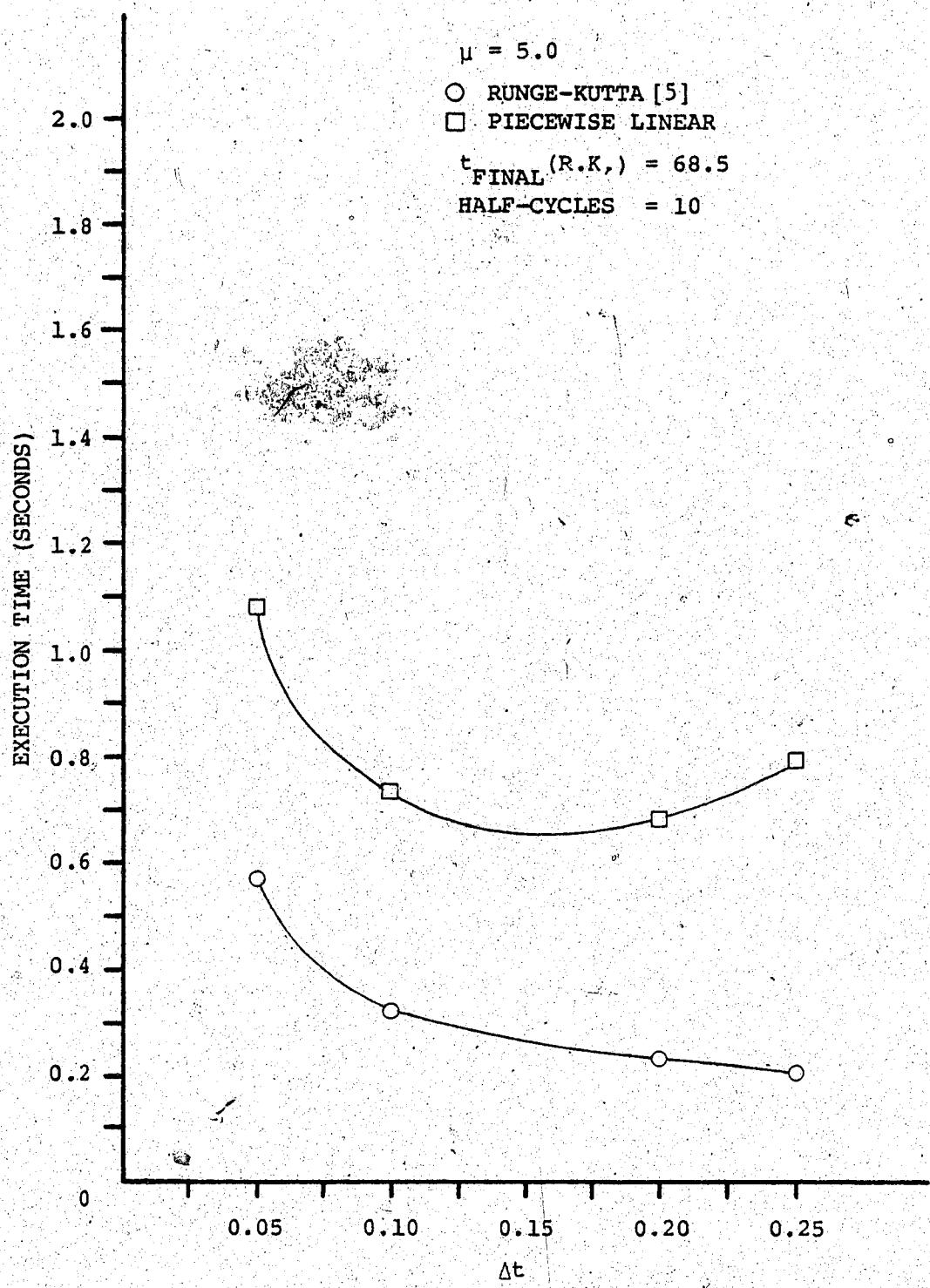


FIGURE 5.26 - EXECUTION TIMES FOR VAN DER POL EQUATION SOLUTIONS FOR $\mu = 5.0$, $x(0) = 3.0$, $\dot{x}(0) = 0.0$ [27]

VI. MATHIEU EQUATION

A. Preliminary Comments

In this chapter, an example of a linear equation with a periodically varying coefficient, namely the Mathieu equation, is examined. It has the canonical form [41]:

$$y'' + (a - 2q \cos 2z)y = 0 , \quad (6.1.1)$$

a = Characteristic number.

The initial conditions considered are:

$$y(0) = y_0, \quad y'(0) = y'_0 . \quad (6.1.2)$$

Fortunately, an exact solution exists in terms of some special functions, and a means of generating a series approximation to them will be derived. This will be used as the basis by which to judge the piecewise linearization. The same Runge-Kutta routine [5] that had been used in the earlier chapters will also be used as confirmation of the results. The investigation will be limited to those cases of integral order.

B. Generation of Characteristic Number

The major references for this chapter are Ince [42] and McLachlan [43].

The appropriate characteristic number depends upon the initial conditions from which the solution is generated, the order of the solution (which determines the period τ) and is summarized as [44]:

$$\tau = \pi \quad \tau = 2\pi$$

$$y(0) = Y_0, \quad y'(0) = 0 \quad a_{2n} \quad a_{2n+1}$$

$$y(0) = 0, \quad y'(0) = Y'_0 \quad b_{2n+2} \quad b_{2n+1}$$

One method for this makes use of the recurrence relations for the coefficients of the series form of the exact solutions [45] which are shown below.

For a_{2n} :

$$ce_{2n}(z, q) = \sum_{r=0}^{\infty} A_{2r}^{(2n)} \cos(2rz) . \quad (6.2.1)$$

For a_{2n+1} :

$$ce_{2n+1}(z, q) = \sum_{r=0}^{\infty} A_{2r+1}^{(2n+1)} \cos(2r + 1)z . \quad (6.2.2)$$

For b_{2n+1} :

$$se_{2n+1}(z, q) = \sum_{r=0}^{\infty} B_{2r+1}^{(2n+1)} \sin(2r + 1)z . \quad (6.2.3)$$

For b_{2n+2} :

$$se_{2n+2}(z, q) = \sum_{r=0}^{\infty} B_{2r+2}^{(2n+2)} \sin(2r + 2)z . \quad (6.2.4)$$

The coefficients for the equations are calculated using the recurrence relations described in McLachlan [46].

For $ce_{2n}(z, q)$:

$$\left. \begin{array}{l} aA_0 - qA_2 = 0, \\ (a - 4)A_2 - q(A_4 + 2A_0) = 0, \\ (a - 4r^2)A_{2r} - q(A_{2r+2} + A_{2r-2}) = 0, \\ \quad (r \geq 2). \end{array} \right\} > (6.2.5)$$

For $ce_{2n+1}(z, q)$:

$$\left. \begin{array}{l} (a - 1 - q)A_1 - qA_3 = 0 \\ [a - (2r + 1)^2]A_{2r+1} \\ - q(A_{2r+3} + A_{2r-1}) = 0, \\ \quad (r \geq 1). \end{array} \right\} > (6.2.6)$$

For $se_{2n+1}(z, q)$:

$$\left. \begin{array}{l} (b - 1 + q)B_1 - qB_3 = 0, \\ [b - (2r + 1)^2]B_{2r+1} \\ - q(B_{2r+3} + B_{2r-1}) = 0, \\ \quad (r \geq 1). \end{array} \right\} > (6.2.7)$$

For $se_{2n+2}(z, q)$:

$$\left. \begin{array}{l} (b - 4)B_2 - qB_4 = 0, \\ (b - 4r^2)B_{2r} \\ - q(B_{2r+2} + B_{2r-2}) = 0, \\ \quad (r \geq 2). \end{array} \right\} > (6.2.8)$$

With Ince's notation [39] for the continued fraction:

$$\Phi \frac{\alpha}{\beta} \nu = \frac{1}{\beta_r} + \frac{\alpha_{r+1}}{\beta_{r+1}} + \frac{\alpha_{r+2}}{\beta_{r+2}} + \dots$$

and the rearrangement of some of the terms, the above relations lead to the following. For $ce_{2n}(z, q)$, the result is [47]:

$$\frac{A_{2r}}{A_{2r-2}} = - \frac{q^2}{4r^2} \frac{\Phi}{\Phi_r} \frac{a}{1 - \frac{a}{4v^2}} - \frac{16v^2(v-1)^2}{1 - \frac{a}{4v^2}}$$

which yields the continued fraction [48]:

$$a_{2n} = - \frac{q^2/2}{1 - \frac{a}{4}} - \frac{q^2/64}{1 - \frac{a}{16}} - \frac{q^2/576}{1 - \frac{a}{36}} - \dots \quad (6.2.9)$$

For $se_{2n+1}(z, q)$, the continued fraction is [49]:

$$\frac{B_{2r-1}}{B_{2r+k}} = \frac{-q}{(2r+1)^2} \frac{\Phi}{\Phi_r} \frac{q^2}{1 - \frac{b}{(2v+1)^2}} - \frac{(4v^2-1)^2}{1 - \frac{b}{(2v+1)^2}}$$

which yields the continued fraction for the characteristic number [49]:

$$b_{2n+1} = 1 - q - \frac{q^2/9}{1 - \frac{b}{9}} - \frac{q^2/225}{1 - \frac{b}{25}} - \frac{q^2/1225}{1 - \frac{b}{49}} - \dots \quad (6.2.10)$$

The case of $ce_{2n+1}(z, q)$ is similar to the previous function since the continued fraction for the coefficients is the same, but with a change of sign in 'q', yielding the value for the characteristic number [49]:

$$a_{2n+1}(q) = b_{2n+1}(-q) \quad (6.2.11)$$

Finally, for $se_{2n+2}(z, q)$ the continued fraction is [49]:

$$\frac{B_{2r}}{B_{2r-2}} = -\frac{q^2}{4r^2} \frac{\infty}{r} \frac{q^2}{16v^2(v-1)^2},$$

and the characteristic number continued fraction is [49]:

$$\begin{aligned} b_{2n+2} &= 4 - \frac{q^2/16}{1 - \frac{b}{16}} - \frac{q^2/576}{1 - \frac{b}{36}} \\ &\quad - \frac{q^2/2304}{1 - \frac{b}{64}} - \dots \end{aligned} \quad (6.2.12)$$

The continued fractions have terms of the form [48]:

$$y_n = \frac{f(a, q, n)}{\phi_{n+1}},$$

$$\phi_{n+1} = f(a, q, n) - y_{n+1}.$$

Calculation is halted when, for the accuracy required, $\phi_{n+1} = 1$.

It was decided that 25 terms would be sufficient.

A characteristic number can now be generated using the continued

fractions just given together with a suggested approach given by Ince [48]. He states how someone doing this task using a calculator, paper and pencil could arrive at a final result. The author took this method and first went through the procedure by hand for tabulated values for the characteristic number and flowcharting the decision processes required; and then testing this method on a Hewlett-Packard HP-67 hand calculator, transferring the final version of the logic to a computer program, which was later run on the Amdahl 470V/8 [26]. The results were compared with the tabulated results given by Ince [50] and there was agreement to seven or eight significant figures.

The following method can be considered as consisting of two parts. The first one (Steps 1-3) is a scan through a range of values for the characteristic number, for an estimated value for a (a_{IN}) and a given q , until an upper and a lower bound, between which the final value will lie, are found. The second one (Steps 4-6) is an interval halving search for this final value. The particular continued fraction used is dependent upon the initial conditions in effect. If at any time during the scan a_{IN} is approximately equal to a_{OUT} , the process stops.

1. A value for a_{OUT} is found using a_{IN} .
2. If a_{IN} is greater than a_{OUT} , assume a_{IN} to be high, decrement it and repeat Step 1 until either a_{IN} is less than a_{OUT} , or the sign of the present a_{OUT} is different than that for the previous a_{OUT} and the present and previous a_{IN} . The upper bound (a_{UP}) is the next-to-last a_{IN} and the lower bound (a_{LOW}) the final one.
3. If a_{IN} is less than a_{OUT} , assume a_{IN} to be low, increment it and repeat Step 1 until a_{IN} is greater than a_{OUT} . The upper bound is the final a_{IN} and the lower one is the previous value.

4. The midpoint a_{MID} is calculated from the average of the upper and lower bounds, and an output value a_{MIDOUT} is found. If a_{MID} is nearly equal to a_{MIDOUT} , a_{MID} is the final value for a . If not, continue below.

5. Consider the case of a positive a_{MID} . If a_{MIDOUT} is less than or equal to zero, a_{LOW} is the present a_{MID} , and Step 4 is repeated. If a_{MIDOUT} is greater than zero and a_{MID} is less than or equal to a_{MIDOUT} , the present a_{MID} becomes a_{LOW} and Step 4 is done again. However, should a_{MIDOUT} be greater than zero, and a_{MID} be greater than a_{MIDOUT} , a_{MID} becomes a_{UP} and the process returns to Step 4.

6. The following applies to a_{MID} less than or equal to zero. If a_{MIDOUT} is greater than zero, a_{MID} becomes the new a_{LOW} , and the next move is to Step 4. For a_{MIDOUT} less than or equal to zero and a_{MID} greater than a_{MIDOUT} , the new a_{UP} is the present a_{MID} , and then to Step 4. If a_{MIDOUT} is less than or equal to zero, and a_{MID} is less than or equal to a_{MIDOUT} , a_{LOW} takes on the value of the present a_{MID} and Step 4 is repeated.

This covers the method for generating a value for a characteristic number. Specifically, the particular calculation chosen depends upon the initial conditions and the proximity of the first guess to the final value.

To elaborate, for a non-zero initial displacement and zero initial velocity, a_{2n} and a_{2n+1} would be the appropriate values, with b_{2n+1} and b_{2b+2} being in effect for zero initial displacement and non-zero initial velocity. The program that was written for this purpose calculated the odd order value first (this choice having been arbitrary) and then the even order one was found. Whichever was closest to a GUESS became the

final result. This was then taken to generate the three solutions used.

C. Series Approximation to Exact Solution

As mentioned earlier, Mathieu functions can be approximated by series solutions [45], which were given as (6.2.1) to (6.2.4). The question now is how to generate the coefficients that were given. According to McLachlan [46], [51], the recurrence relations that were given in (6.2.5) to (6.2.8) can be used. However, a more suitable form for these relations must be obtained in order to facilitate calculation of the coefficients.

Consider $ce_{2n}(z, q)$ [52]. The first relation of (6.2.5) gives:

$$\frac{A_2}{A_0} = \frac{a}{q}$$

The next one yields:

$$\frac{A_4}{A_2} = \frac{a - 4}{q} - \frac{2}{A_2/A_0}$$

The final one gives:

$$\frac{A_{2r}}{A_{2r-2}} = \frac{a}{q} - \frac{a - 4r^2 - q \left[\frac{A_{2r+2}}{A_{2r}} \right]}{(r \geq 3)}$$

It had been decided to terminate calculation after 25 terms, since ratios beyond A_{52}/A_{50} would become so small that they could be neglected.

when compared with the other terms [53]. Once the various ratios have been found, the normalization [54]:

$$1 = 2A_0^2 + A_2^2 + A_4^2 + \dots$$

$$\frac{1}{A_0^2} = 2 + \left(\frac{A_2}{A_0}\right)^2 + \left(\frac{A_4}{A_0}\right)^2 + \dots$$

is required, and so the values for A_{2r}/A_0 are needed. This is done by using:

$$\frac{A_{2r}}{A_0} = \frac{A_{2r}}{A_{2r-2}} \cdot \frac{A_{2r-2}}{A_0}$$

starting with A_2/A_0 and A_4/A_2 , and working sequentially through all the ratios. Once this has been accomplished, using the different values for A_{2r}/A_0 , A_0 and A_{2r} can be found.

With the various A_{2r} 's, a series approximation for $ce_{2n}(z, q)$ can be generated.

Take now $ce_{2n+1}(z, q)$ [55]. The first ratio from (6.2.6) gives:

$$\frac{A_3}{A_1} = \frac{a - 1 - q}{q}$$

The remaining ratios are of the form:

$$\frac{A_{2r+1}}{A_{2r-1}} = \frac{q}{a - (2r + 1)^2 - q \left[\frac{A_{2r+3}}{A_{2r+1}} \right]}$$

($r \geq 2$)

As was the case for the previous function, calculation was halted

after 25 terms because ratios beyond A_{51}/A_{49} can be considered as being negligible when compared with the others. The resulting normalization used is [56]:

$$1 = A_1^2 + A_3^2 + A_5^2 + \dots ,$$

and the series approximation for $ce_{2n+1}(z, q)$ using the values for A_{2r+1} is found using the same method as before.

Consider now $se_{2n+1}(z, q)$. Owing to the similarities between (6.2.6) and (6.2.7), the derivations can be omitted and the results presented [49]:

$$\frac{B_3}{B_1} = \frac{b - 1 + q}{q},$$

$$\frac{B_{2r+1}}{B_{2r-1}} = \frac{q}{b - (2r + 1)^2 - q \left[\frac{B_{2r+3}}{B_{2r-1}} \right]}$$

$$(r \geq 2).$$

$$1 = B_1^2 + B_3^2 + B_5^2 + \dots .$$

From this, $se_{2n+1}(z, q)$ can be found since the coefficients can be determined from the above relations.

Finally, the results for $se_{2n+2}(z, q)$ will be shown. Since (6.2.8) is similar to (6.2.5), the ratios are as follows, since the derivations would be alike [55], [56]:

$$\frac{B_4}{B_2} = \frac{b - 4}{q},$$

$$\frac{B_{2r}}{B_{2r-2}} = \frac{q}{b - 4r^2 - q \left[\frac{B_{2r+2}}{B_{2r}} \right]} ,$$

($r \geq 3$).

$$1 = B_2^2 + B_4^2 + B_6^2 + \dots .$$

From this, an approximation to a Mathieu function of integral order can be generated, thus allowing one to consider the complete solution to the original equation. A computer program was written to make use of the relations just derived, and the results were checked against tabulated values that Ince presented [57], and were found to be in agreement to at least seven significant figures in almost all cases.

Using these coefficients, (6.2.1) to (6.2.4) can now be calculated, and thus, an approximation to the exact solution generated. For integral order solutions, one or the other of (6.1.2) is zero. For:

$$y(0) = Y_0 ,$$

$$y'(0) = 0 ,$$

the solution uses either (6.2.1) or (6.2.2), giving a complete result of:

$$y = \alpha c e_{\phi(r)}(z, q)$$

$$= \alpha \sum_{r=0}^{\infty} A_{\phi(r)} \cos [\phi(r)z] ,$$

where:

$$\begin{aligned}\phi(r) &= 2r, & (\tau = \pi) \\ &= 2r+1, & (\tau = 2\pi)\end{aligned}$$

Applying (6.1.2) yields the final form:

$$y_0 = \alpha \sum_{r=0}^{\infty} A_r \phi(r) \cos(0),$$

$$\alpha = \frac{Y_0}{\sum_{r=0}^{\infty} A_r \phi(r)},$$

$$y = \frac{Y_0}{\sum_{r=0}^{\infty} A_r \phi(r)} \sum_{r=0}^{\infty} A_r \phi(r) \cos[\phi(r)z].$$

For:

$$y(0) = 0,$$

$$y'(0) = Y_0,$$

the solution is of the form of either (6.2.3) or (6.2.4), thus yielding:

$$\begin{aligned}y &= \beta s e_{\phi(r)}(z, q) \\ &= \beta \sum_{r=0}^{\infty} B_r \phi(r) \sin[\phi(r)z],\end{aligned}$$

with:

$$\begin{aligned}\phi(r) &= 2r+2, & (\tau = \pi) \\ &= 2r+1, & (\tau = 2\pi)\end{aligned}$$

giving, after using (6.1.2):

$$y' = \beta \sum_{r=0}^{\infty} \{ \phi(r) B_{\phi(r)} \cos [\phi(r)z] \}$$

$$y'_0 = \beta \sum_{r=0}^{\infty} [\phi(r) B_{\phi(r)} \cos(0)]$$

$$\beta = \frac{y'_0}{\sum_{r=0}^{\infty} \phi(r) B_{\phi(r)}} ,$$

$$y = \frac{y'_0}{\sum_{r=0}^{\infty} \phi(r) B_{\phi(r)}} \sum_{r=0}^{\infty} B_{\phi(r)} \sin [\phi(r)z] .$$

D. Derivation of Piecewise Linear Solution

Since it is desired to have a piecewise linear solution of the form shown in (1.2), (6.2.1) can be approximated by:

$$y'' + p^2 y = 0$$

over a small interval, Δz . The displacement at the beginning of this interval is y_0 and the velocity y'_0 .

This equation has the following solutions (based on [34]):

A. $p^2 > 0$ [58]:

$$y = y'_0 \cos pz + \frac{y'_0}{p} \sin pz .$$

B. $p^2 = 0$ [37]:

$$y = y'_0 z + y_0 .$$

c. $p^2 < 0$ [37]:

$$y = y_0 \cosh sz + \frac{y'_0}{s} \sinh sz ,$$

$$p^2 = -s^2 .$$

The final value for the characteristic number, a, and q are given such that the solution would be periodic. Δz can be of any reasonable size.

Consider now the $(m - 1)$ th and mth points. The displacements that would correspond to them are y_{m-1} and y_m , respectively. See Figure 6.1 for details. As the system moves through Δz , p^2 will change. It is better to have it as a constant value as it passes through an interval, and so, it is calculated at the midpoint of the interval, or at $(m - 1/2)\Delta z$. It is obvious that this will be the case as Δz gets smaller. That is to say, $a - 2q\cos[2(m-1)\Delta z]$ and $a - 2q\cos(2m\Delta z)$ will approach $a - 2q\cos[(2m-1)\Delta z]$.

So, for the mth point, the solutions will be as follows. For $a - 2q\cos[(2m-1)\Delta z]$ greater than zero [58]:

$$y_m = y_0 \cos p_m \Delta z + \frac{y'_0}{p_m} \sin p_m \Delta z ,$$

$$p_m = [a - 2q \cos (2m - 1) \Delta z]^{1/2} .$$

For $a - 2q\cos[(2m-1)\Delta z]$ equal to zero, the solution becomes [37]:

$$y_m = y'_0 \Delta z + y_0 .$$

And finally, for $a - 2q\cos[(2m-1)\Delta z]$ less than zero [37]:

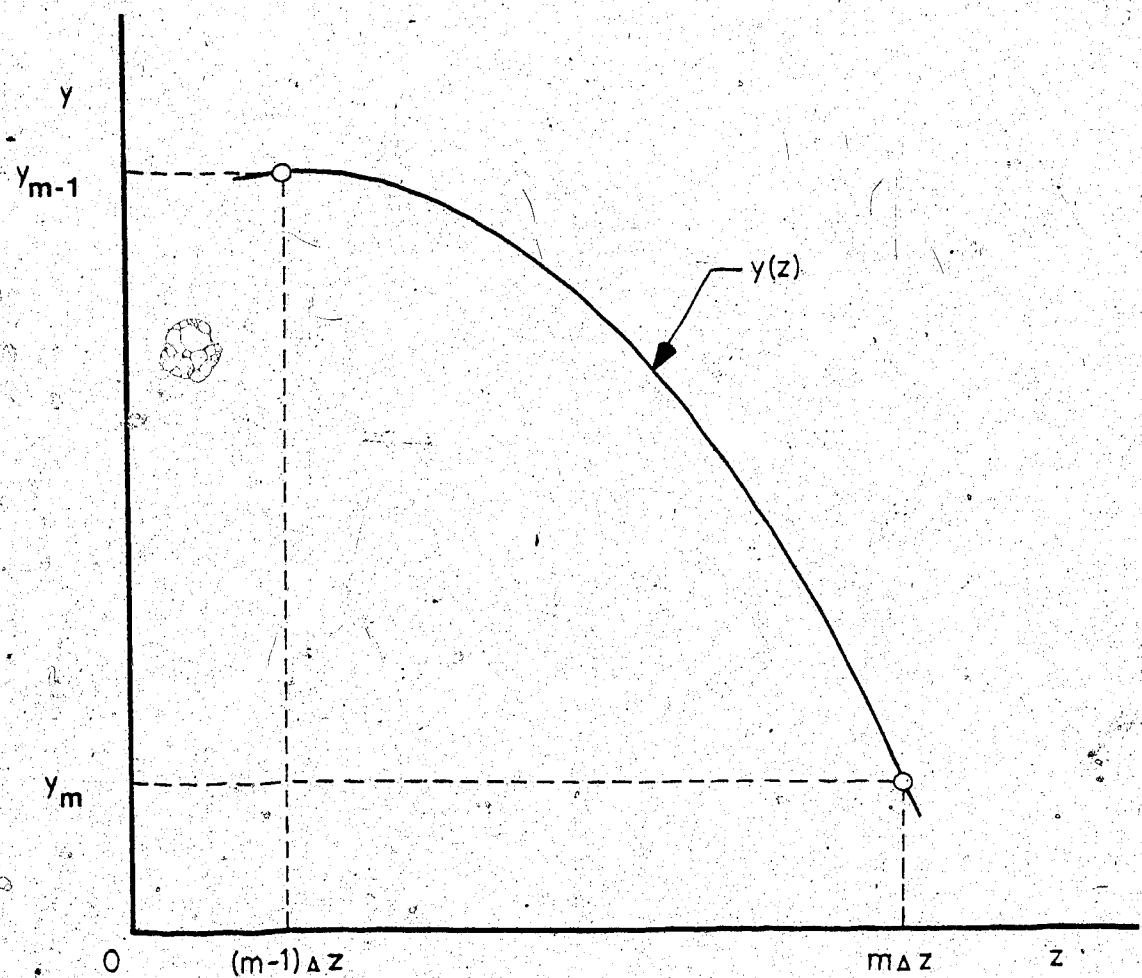


FIGURE 6.1 - TYPICAL INTERVAL LAYOUT (MATHIEU EQUATION)

$$y_m = y_0 \cosh s_m \Delta z + \frac{y'_0}{s_m} \sinh s_m \Delta z ,$$

$$s_m = \{-[a - 2q \cos(2m - 1)\Delta z]\}^{1/2} .$$

For each successive interval,

$$y_0 = y_{m-1} ,$$

$$y'_0 = y'_{m-1} .$$

At the end of the interval:

$$y_0 = y_m ,$$

$$y'_0 = y'_m .$$

and the next set of calculations for finding y_m can begin.

E. Results

Since there are numerous combinations of the characteristic number, q , and initial conditions that one can examine, it was decided to take a few selected values for the parameters and initial conditions, which would illustrate what can be encountered.

It was decided to take the initial guesses for the characteristic number as -50.0, 10.0, and 30.0, with the values for q being 1.0, 10.0, 20.0, 30.0, and 40.0. This enabled one to check some of the values obtained against tabulated results.

For the case of $q = 1.0$, both the piecewise linearization and the Runge-Kutta solution [5] had a close correspondence with the series

approximation to the exact solution, as can be seen in Figures 6.2 - 6.4. For most of the values, the same applies for $q = 10.0$ (Figures 6.5 and 6.6 being examples of the results obtained). However, something unusual occurred for the case of $q = 10.0$, a $\text{GUESS} = -50.0$, $y(0) = 0.0$, $y'(0) = 1.0$ (Figure 6.7), where the piecewise linearization exhibits erratic behaviour.

Consider the cases where $q = 20.0$. Figure 6.8 is a typical example of the good correspondence between all three solutions, but Figure 6.9 shows some anomalous results. In fact, tests for this case have shown that the amplitude of the piecewise linear results can go as high as 5×10^5 , thus making the Runge-Kutta solution [5] and series approximation insignificant by comparison. A somewhat minor deviation was found to exist for the same values of a GUESS and q , but for the other set of initial conditions, though this occurs near the end of the first cycle, and was less than a third of the maximum values of the other solutions. Up to that point, however, it had given satisfactory results.

Since this has been observed to happen, the question now is whether this occurs for other values of a GUESS and q . Indeed it does. Compare the difference between a "good" result (that is, correspondence between all three solutions) and one that is not (Figures 6.10 and 6.11, respectively, for $q = 30.0$, and Figures 6.12 and 6.13, respectively, for $q = 40.0$).

The magnitude and form of the deviations was totally unexpected, since they did not even follow the basic pattern of the other solutions, but seem to be random occurrences, quite unlike what was seen in the previous chapter where the shape of the resulting piecewise linear solution curve at least was the same as that of the solution being

FIGURE 6.2 - SOLUTIONS TO MATHIEU EQUATION FOR

$a_{\text{GUESS}} = -50.0$, $q = 1.0$, $\Delta z = \pi/200.00$,
 $y(0) = 0.0$, $y'(0) = 1.0$

(N.B.: Runge-Kutta solution uses [5].)

CHAR. NUM. (EST. VALUE) = -50.0000
CHAR. NUM. (FINAL VALUE) = -0.1102
 $q = 1.0000 \quad \Delta z = \pi/200.0$

$$y(0) = 0.0000 \quad y'(0) = 1.0000$$

— SERIES APPROXIMATION
○ PIECEWISE LINEARIZATION
△ RUNGE-KUTTA

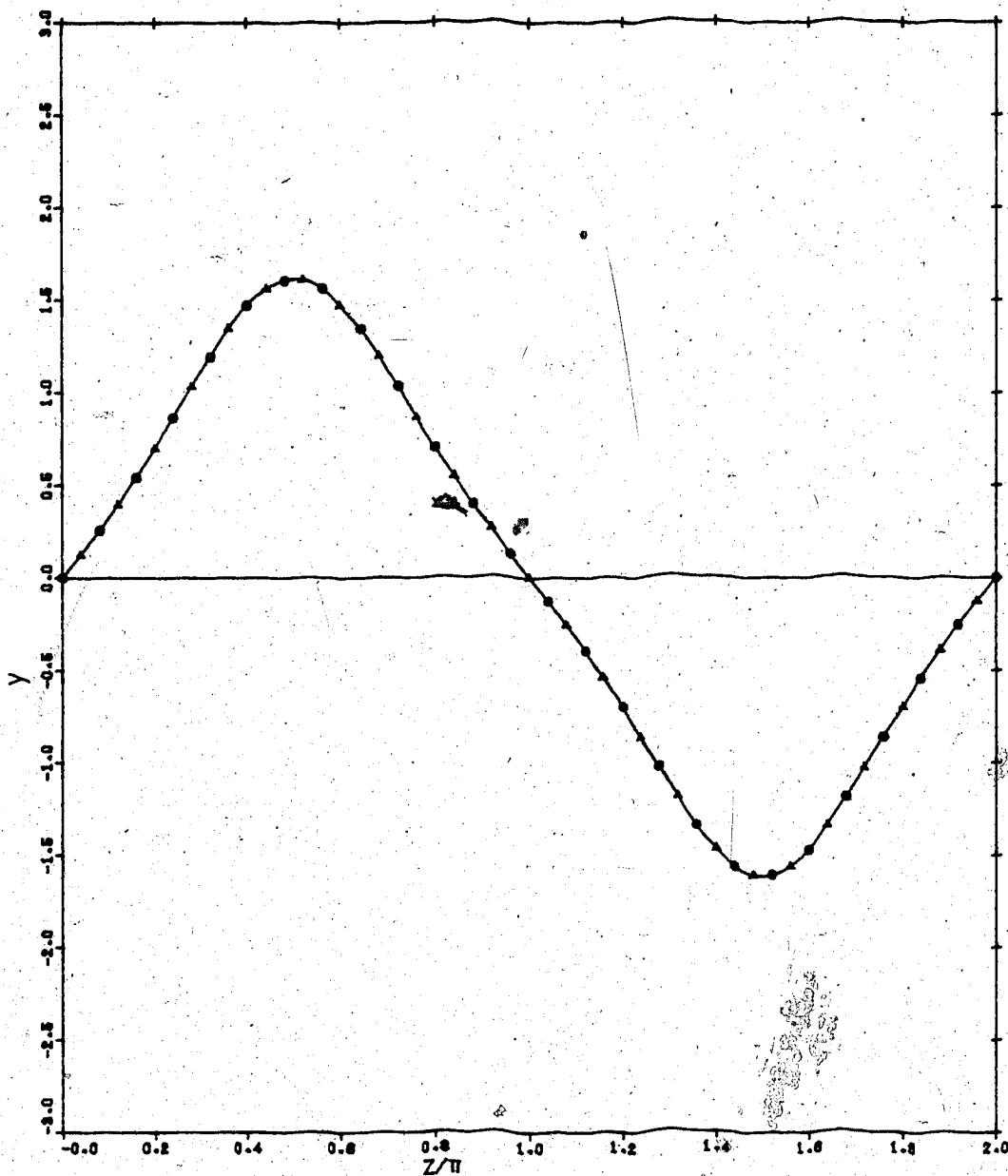


FIGURE 6.3 - SOLUTIONS TO MATHIEU EQUATION FOR

$a_{\text{GUESS}} = 10.0$, $q = 1.0$, $\Delta z = \pi/200.00$,
 $y(0) = 1.0$, $y'(0) = 1.0$

(N.B.: Runge-Kutta solution uses [5].)

CHAR. NUM. (EST. VALUE) = 10.0000
CHAR. NUM. (FINAL VALUE) = 9.0784
 $q = 1.0000 \quad \Delta z = \pi/200.0$

$$y(0) = 1.0000 \quad y'(0) = 0.0000$$

— SERIES APPROXIMATION
○ PIECEWISE LINEARIZATION
△ RUNGE-KUTTA

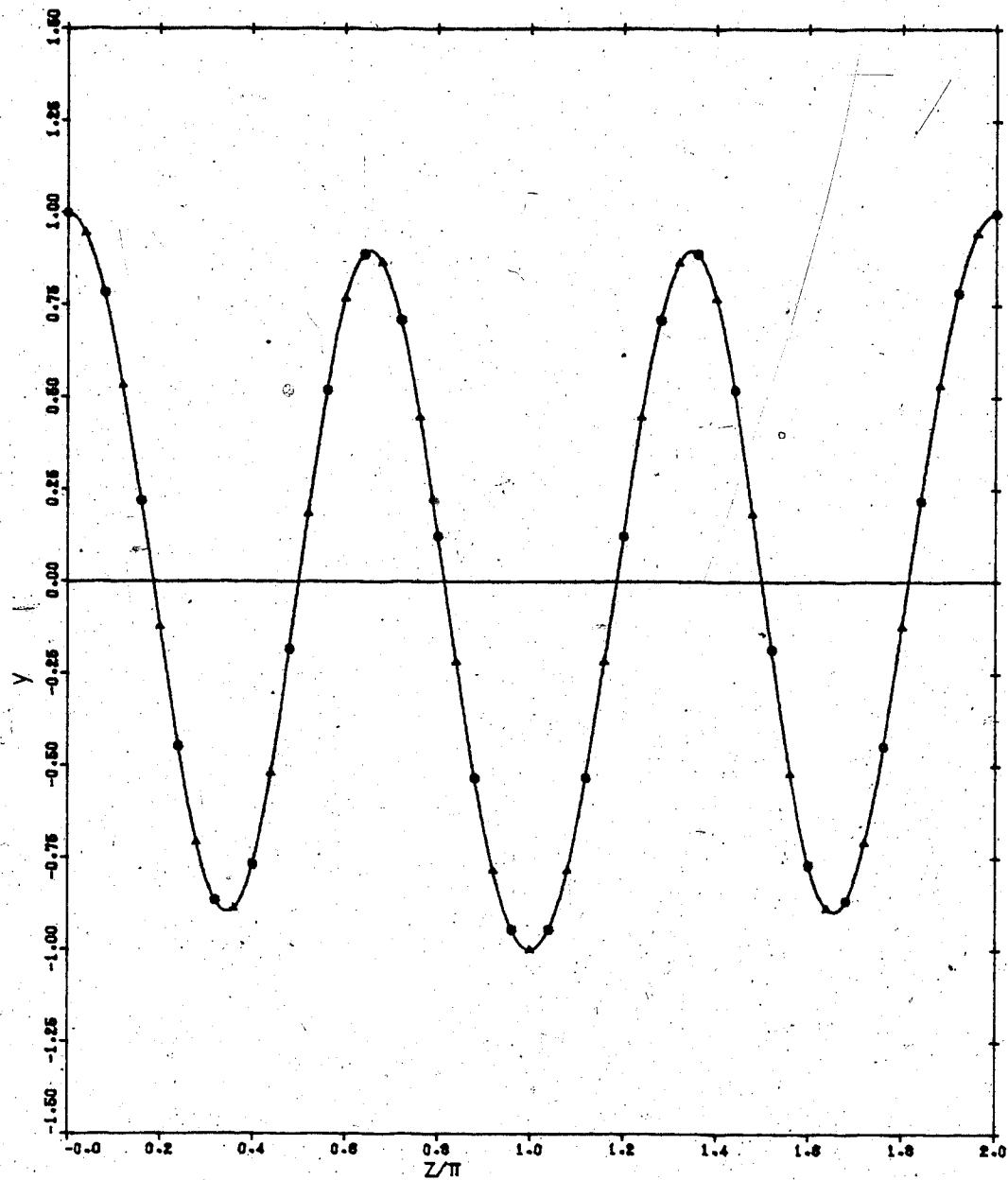


FIGURE 6.4 - SOLUTIONS TO MATHIEU EQUATION FOR

$a_{\text{GUESS}} = 30.0$, $q = 1.0$, $\Delta z = \pi/200.00$,
 $y(0) = 0.0$, $y'(0) = 1.0$

(N.B.: Runge-Kutta solution uses [5].)

CHAR. NUM. (EST. VALUE) = 30.0000
CHAR. NUM. (FINAL VALUE) = 25.0208
 $q = 1.0000 \quad \Delta z = \pi/200.0$

$$y(0) = 0.0000 \quad y'(0) = 1.0000$$

— SERIES APPROXIMATION
○ PIECEWISE LINEARIZATION
△ RUNGE-KUTTA

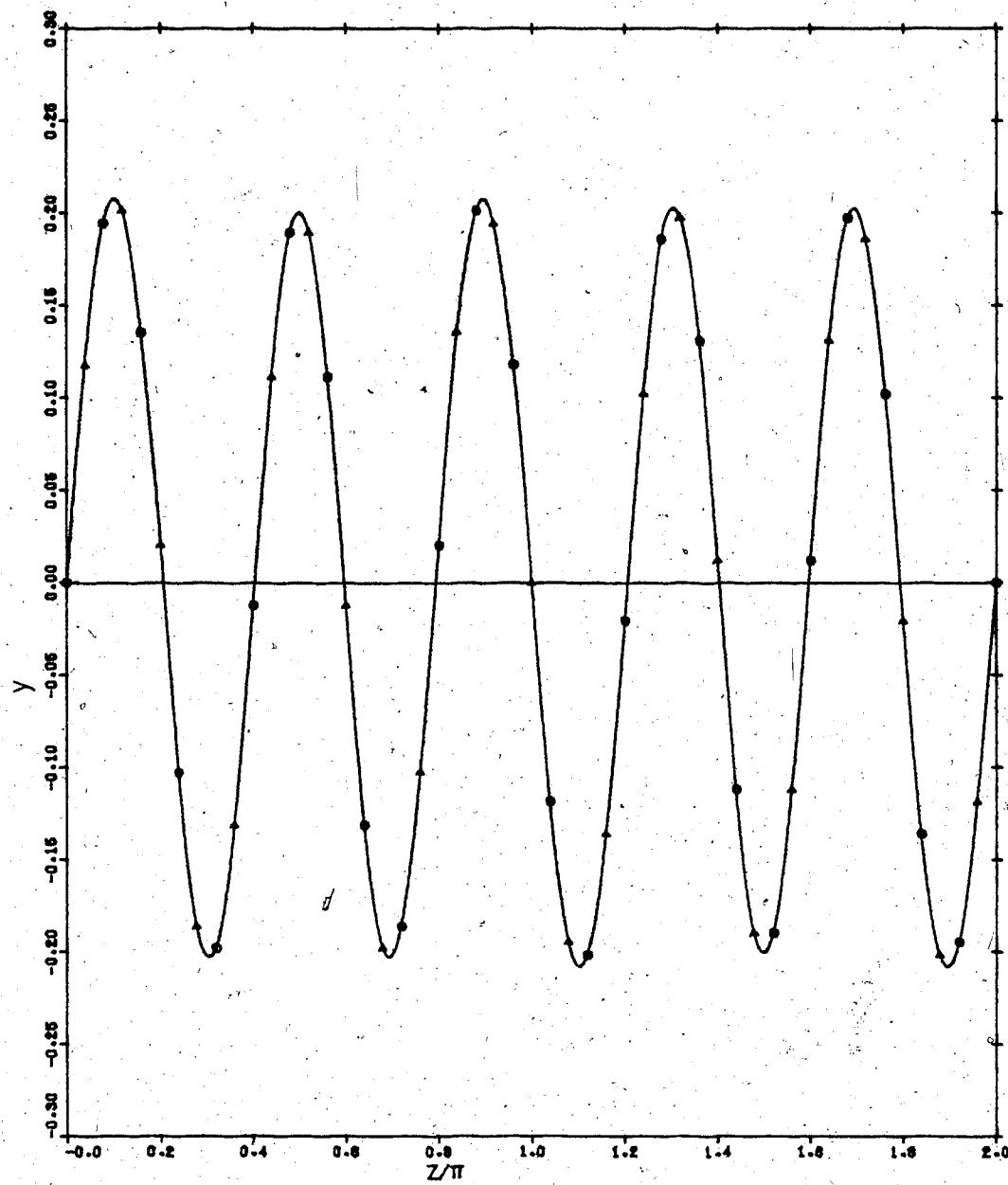


FIGURE 6.5 - SOLUTIONS TO MATHIEU EQUATION FOR

a = 10.0, q = 10.0, $\Delta z = \pi/200.00$,
GUESS $y(0) = 0.0, y'(0) = 1.0$

(N.B.: Runge-Kutta solution uses [5].)

CHAR. NUM. (EST. VALUE) = 10.0000

CHAR. NUM. (FINAL VALUE) = 7.9861

$q = 10.0000 \quad \Delta z = \pi/200.0$

$y(0) = 0.0000 \quad y'(0) = 1.0000$

— SERIES APPROXIMATION
○ PIECEWISE LINEARIZATION
△ RUNGE-KUTTA

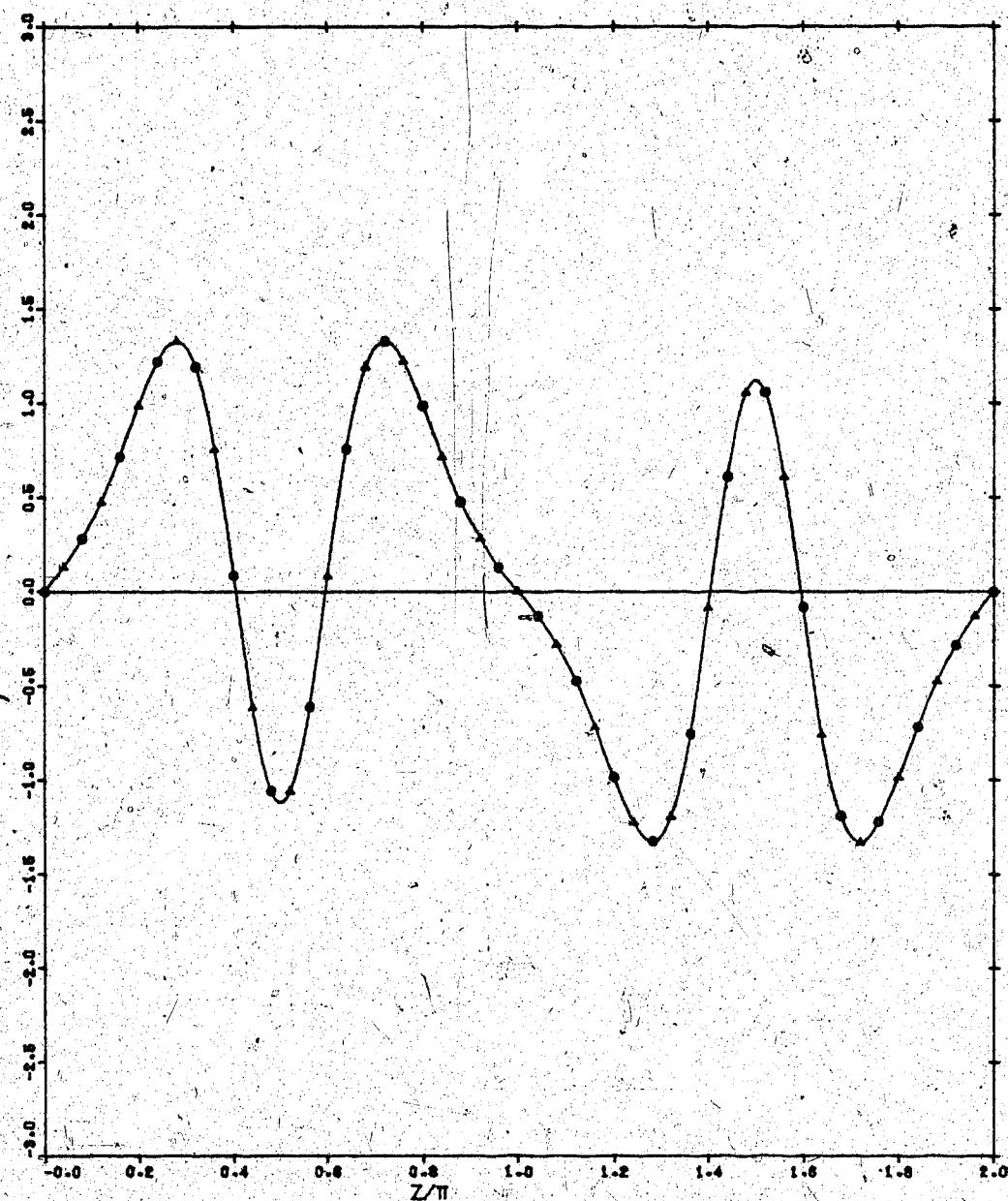


FIGURE 6.6 - SOLUTIONS TO MATHIEU EQUATION FOR
 $a_{\text{GUESS}} = 30.0$, $q = 10.0$, $\Delta z = \pi/200.00$,
 $y(0) = 0.0$, $y'(0) = 1.0$

(N.B.: Runge-Kutta solution uses [5].)

CHAR. NUM. (EST. VALUE) = 30.0000
CHAR. NUM. (FINAL VALUE) = 26.7664
 $q = 10.0000 \quad \Delta z = \pi/200.0$
 $y(0) = 0.0000 \quad y'(0) = 1.0000$

— SERIES APPROXIMATION
○ PIECEWISE LINEARIZATION
△ RUNGE-KUTTA

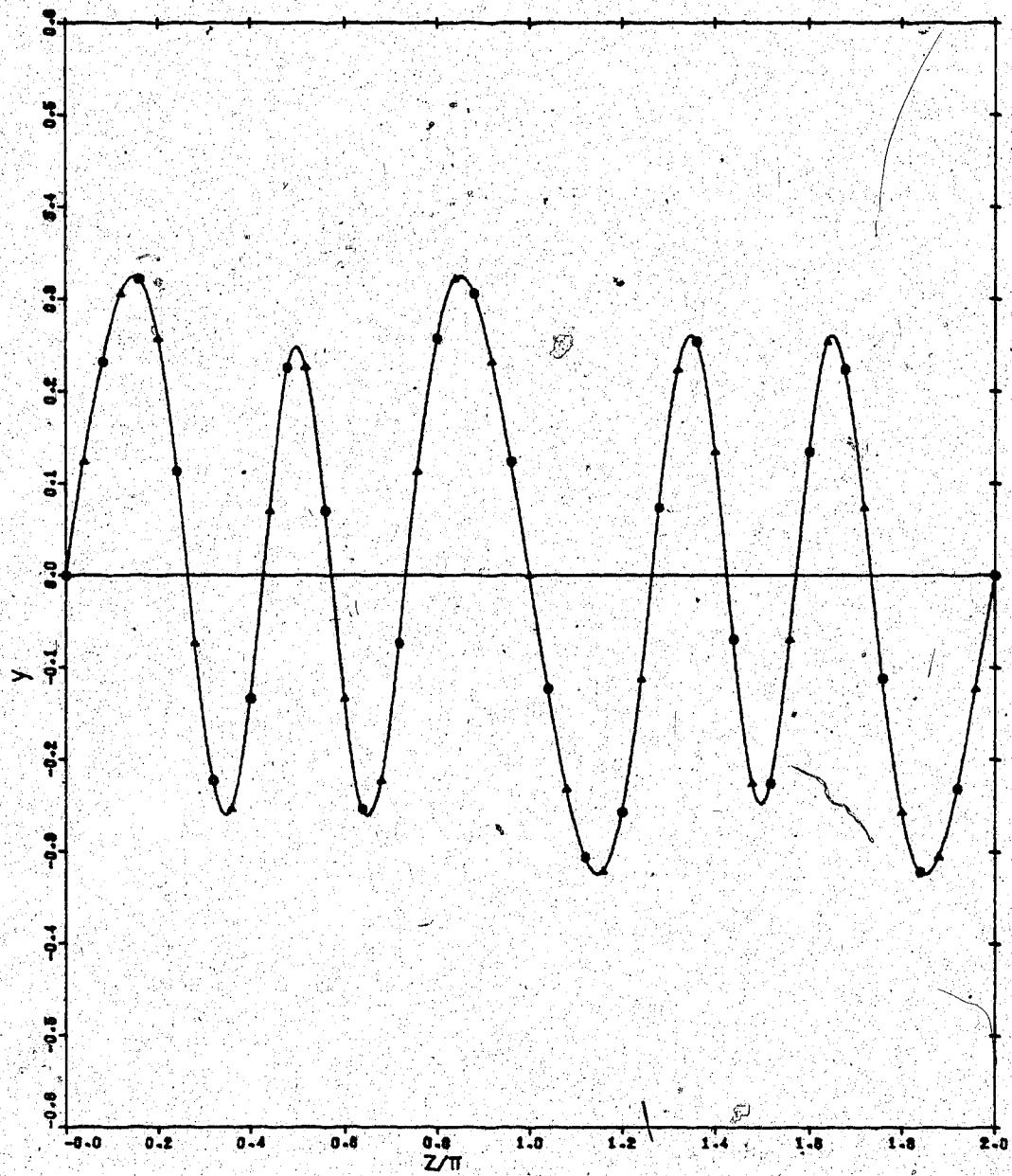


FIGURE 6.7 - SOLUTIONS TO MATHIEU EQUATION FOR
a = -50.0, q = 10.0, $\Delta z = \pi/200.00$,
GUESS y(0) = 0.0, $y'(0) = 1.0$

(N.B.: Runge-Kutta solution uses [5].)

CHAR. NUM. (EST. VALUE) = -50.0000
CHAR. NUM. (FINAL VALUE) = -13.9365
 $q = 10.0000 \quad \Delta z = \pi/200.0$

$y(0) = 0.0000, \quad y'(0) = 1.0000$

— SERIES APPROXIMATION
○ PIECEWISE LINEARIZATION
△ RUNGE-KUTTA

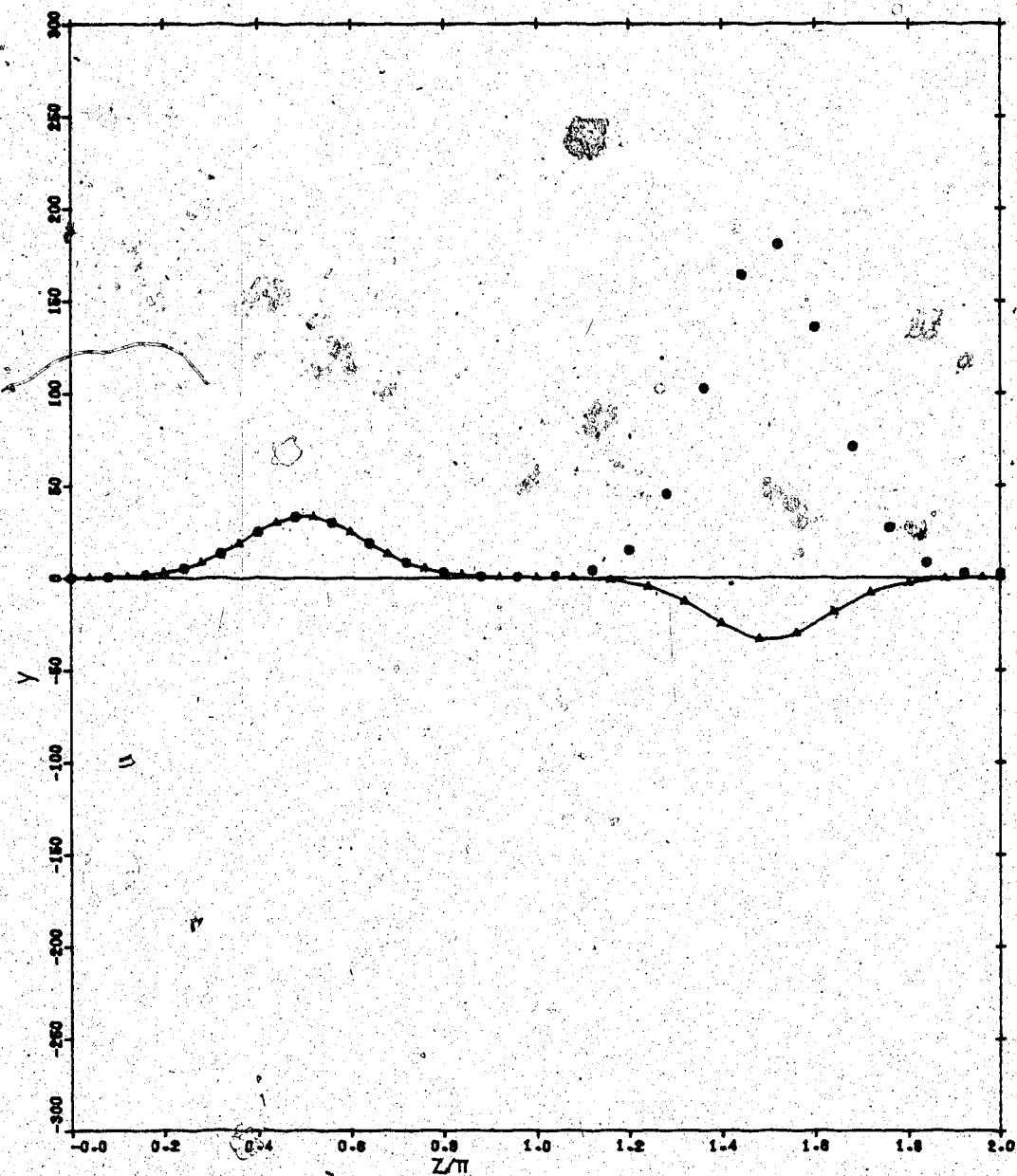


FIGURE 6.8 - SOLUTIONS TO MATHIEU EQUATION FOR
aGUESS = 10.0, q = 20.0, Δz = π/200.00,
y(0) = 1.0, y'(0) = 0.0

(N.B.: Runge-Kutta solution uses [5].)

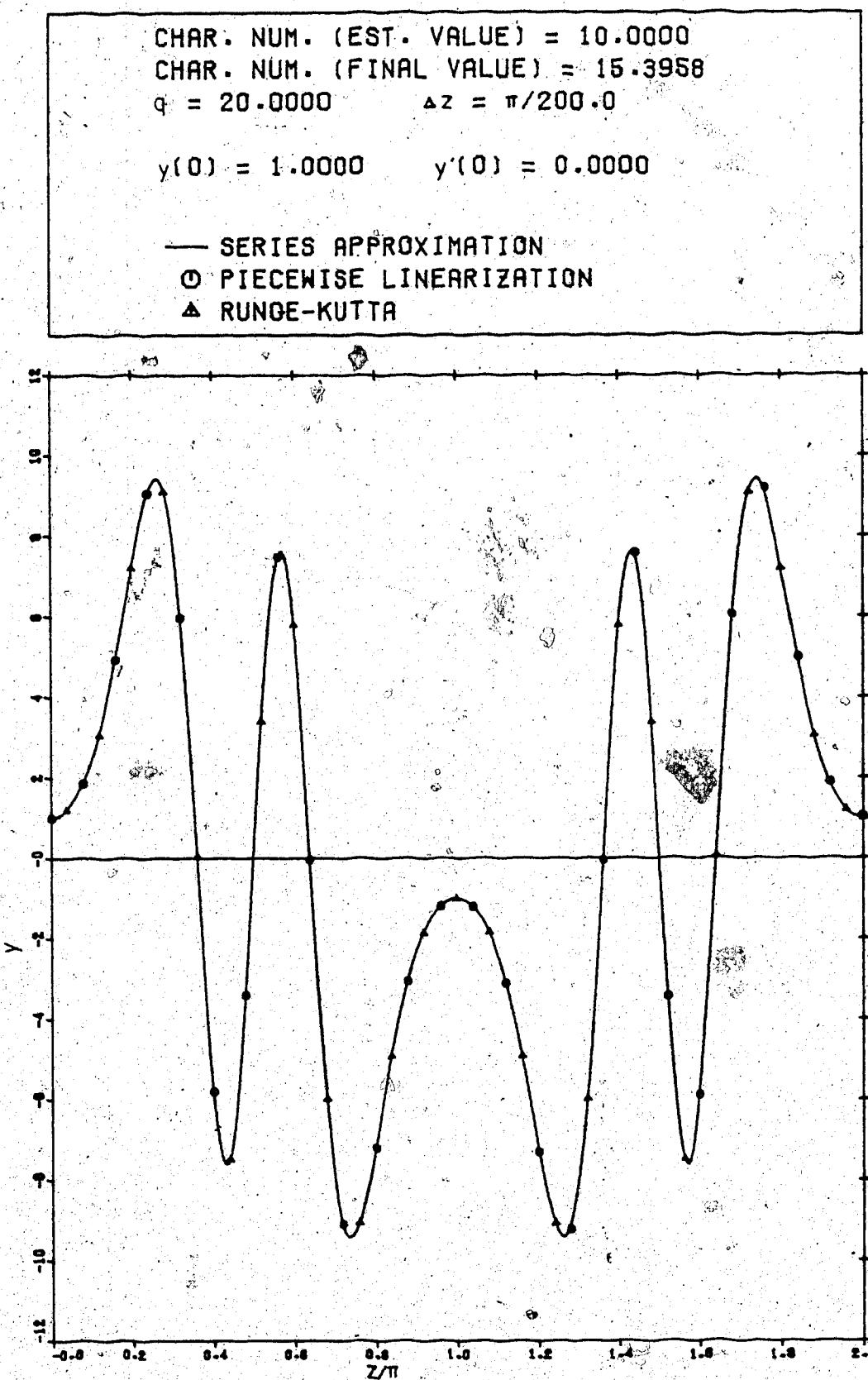


FIGURE 6.9 - SOLUTIONS TO MATHIEU EQUATION FOR

$a = -50.0, q = 20.0, \Delta z = \pi / 200.00,$
GUESS $y(0) = 0.0, y'(0) = 1.0$

(N.B.: Runge-Kutta solution uses [5].)

CHAR. NUM. (EST. VALUE) = -50.0000
CHAR. NUM. (FINAL VALUE) = -31.3134
 $q = 20.0000 \quad \Delta z = \pi/200.0$

$$y(0) = 0.0000 \quad y'(0) = 1.0000$$

— SERIES APPROXIMATION
○ PIECEWISE LINEARIZATION
△ RUNGE-KUTTA

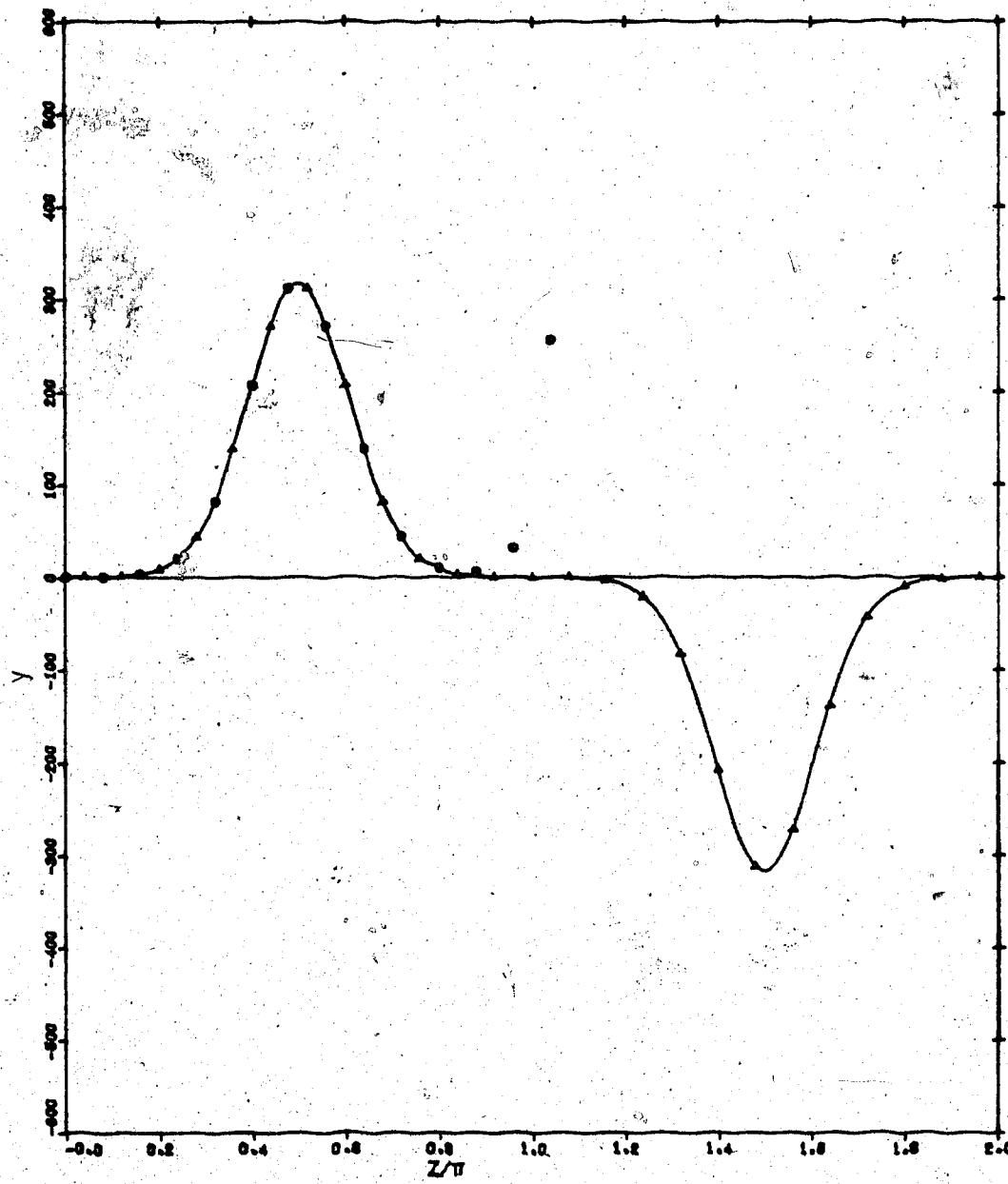


FIGURE 6.10 - SOLUTIONS TO MATHIEU EQUATION FOR

$a_{\text{GUESS}} = 30.0$, $q = 30.0$, $\Delta z = \pi/200.00$,
 $y(0) = 0.0$, $y'(0) = 1.0$

(N.B.: Runge-Kutta solution uses [5].)

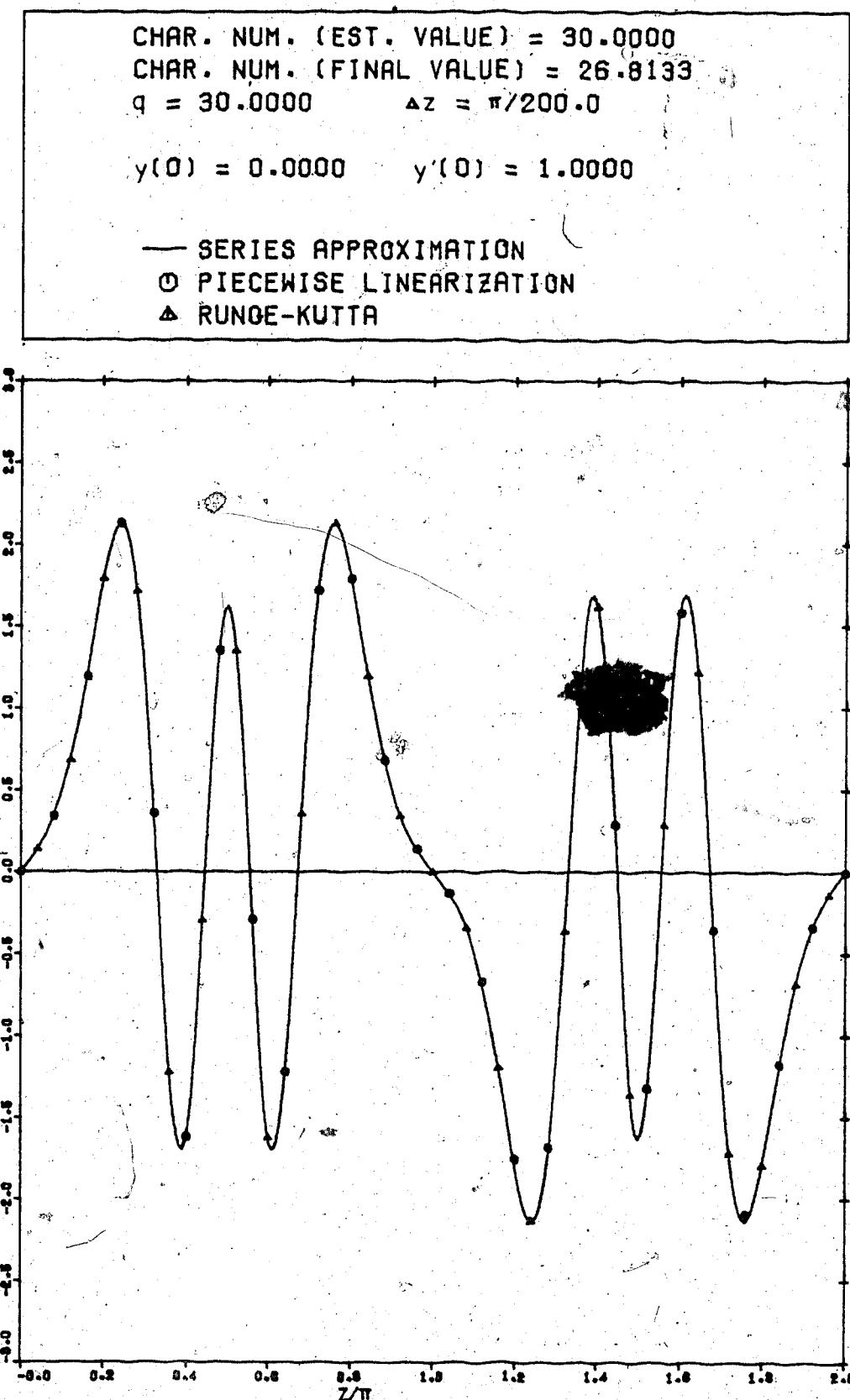


FIGURE 6.11 - SOLUTIONS TO MATHIEU EQUATION FOR
 $a_1 = -50.0$, $q = 30.0$, $\Delta z = \pi/200.00$,
GUESS $y(0) = 0.0$, $y'(0) = 1.0$

(N.B.: Runge-Kutta solution uses [5].)

CHAR. NUM. (EST. VALUE) = -50.0000
CHAR. NUM. (FINAL VALUE) = -49.3016
 $q = 30.0000 \quad \Delta z = \pi/200.0$

$$y(0) = 0.0000 \quad y'(0) = 1.0000$$

— SERIES APPROXIMATION
○ PIECEWISE LINEARIZATION
△ RUNGE-KUTTA

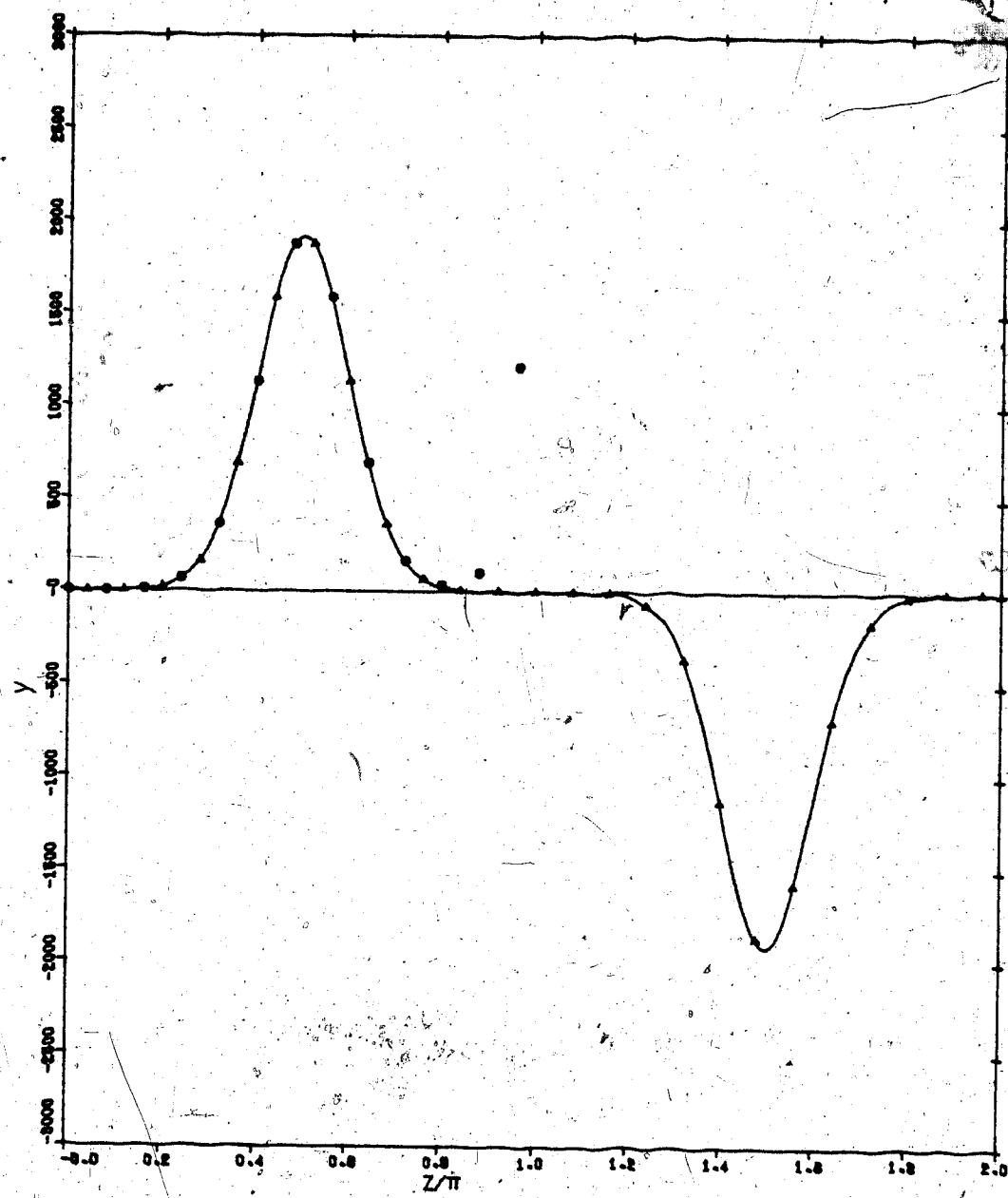


FIGURE 6.12 - SOLUTIONS TO MATHIEU EQUATION FOR
a_{GUESS} = 30.0, q = 40.0, Δz = π/200.00,
y(0) = 1.0, y'(0) = 0.0

(N.B.: Runge-Kutta solution uses [5].)

CHAR. NUM. (EST. VALUE) = 30.0000
CHAR. NUM. (FINAL VALUE) = 22.3253
 $q = 40.0000 \quad \Delta z = \pi/200.0$
 $y(0) = 1.0000 \quad y'(0) = 0.0000$

— SERIES APPROXIMATION
○ PIECEWISE LINEARIZATION
△ RUNGE-KUTTA

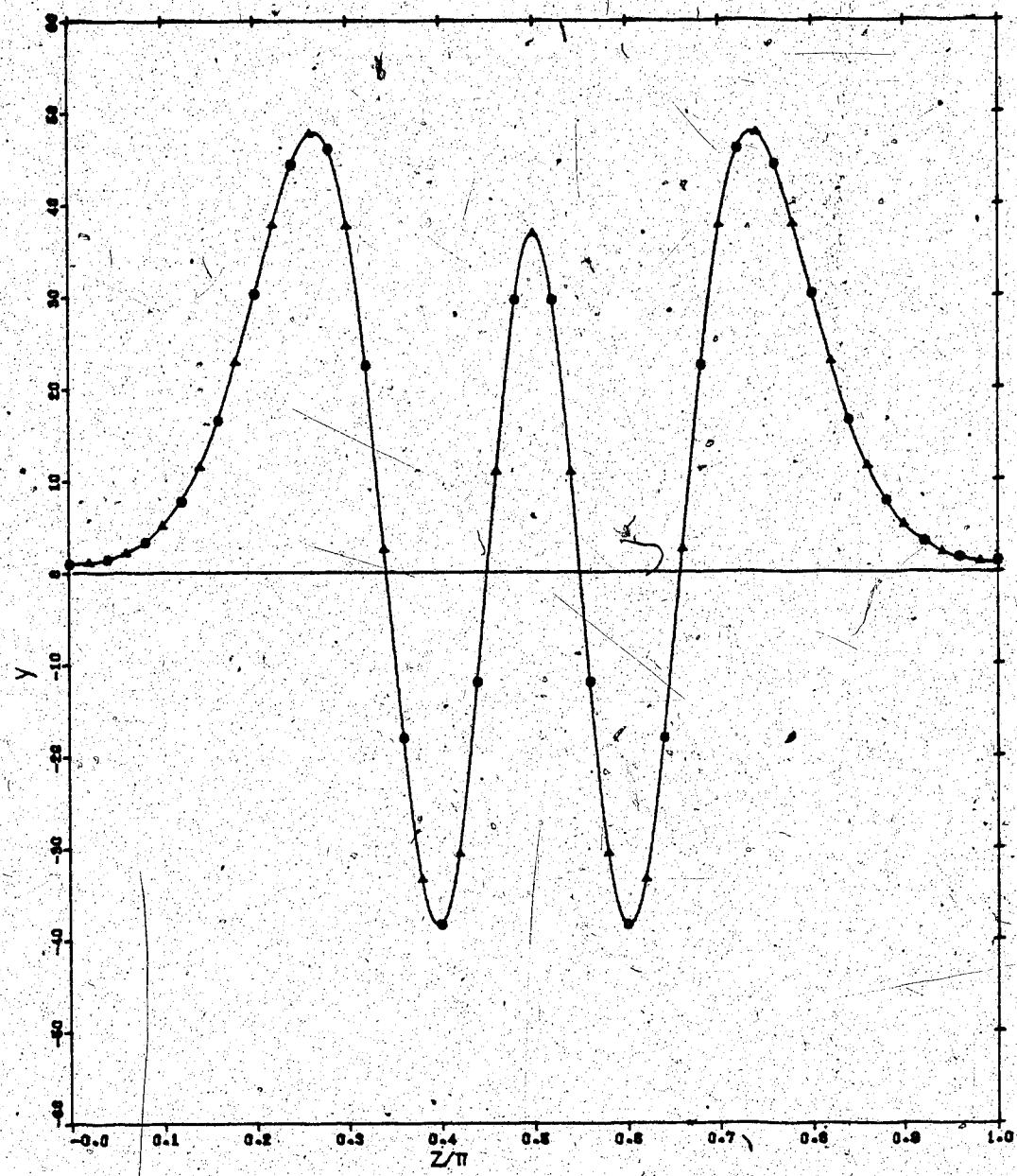


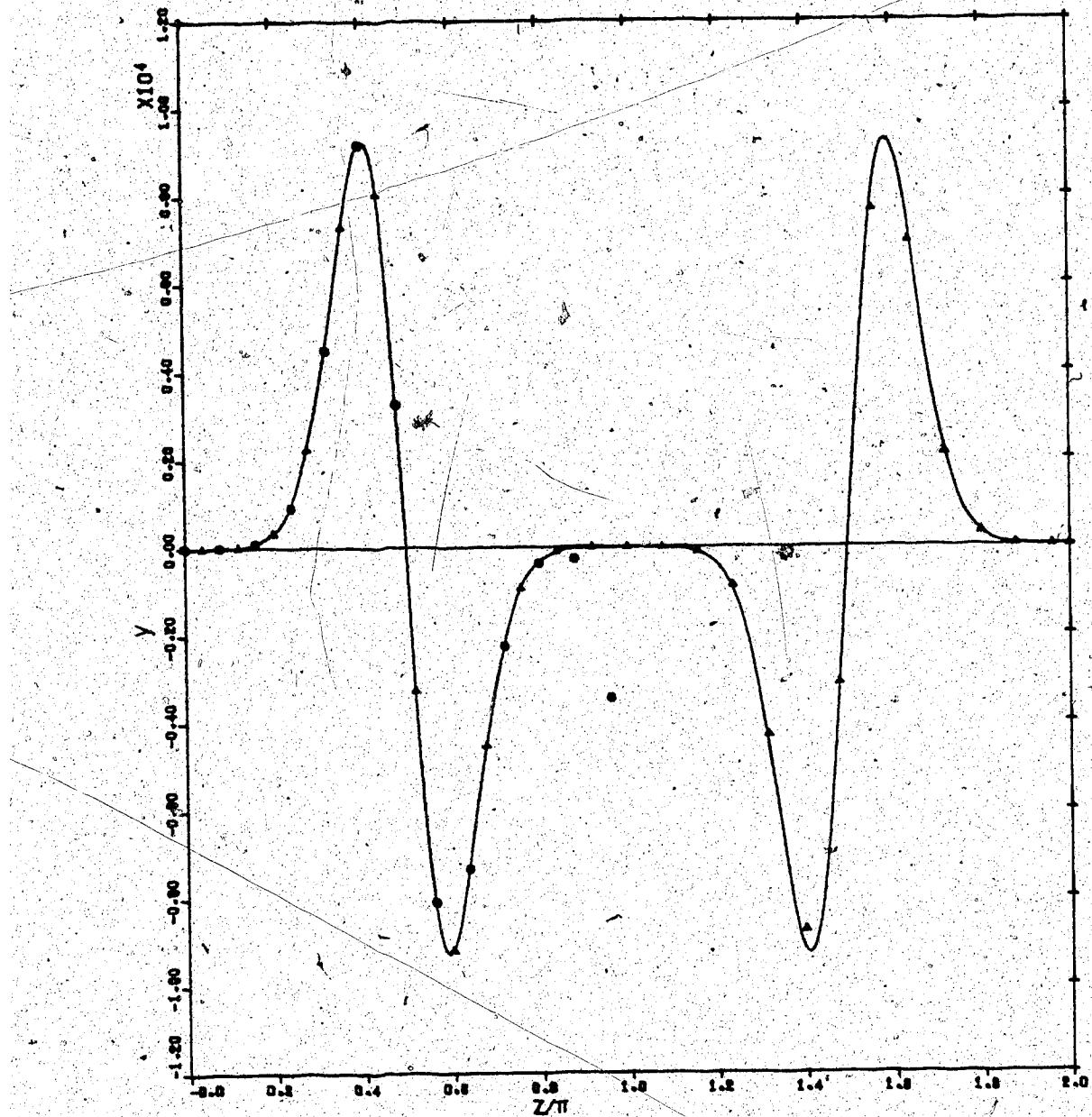
FIGURE 6.13 - SOLUTIONS TO MATHIEU EQUATION FOR
 $a = -50.0$, $q = 40.0$, $\Delta z = \pi/200.00$,
GUESS $y(0) = 1.0$, $y'(0) = 0.0$.

(N.B.: Runge-Kutta Solution uses [5].)

CHAR. NUM. (EST. VALUE) = -50.0000
CHAR. NUM. (FINAL VALUE) = -43.3522
 $q = 40.0000 \quad \Delta z = \pi/200.0$

$$y(0) = 1.0000 \quad y'(0) = 0.0000$$

— SERIES APPROXIMATION
○ PIECEWISE LINEARIZATION
△ RUNGE-KUTTA



compared. Since changing the size of the increment of z seemed to work for the Van der Pol equation, it was felt that this may solve the problem. A typical result of that was as can be seen in Figure 6.14. This particular case was chosen as this phenomenon was first observed during the initial development of the computer program using the same conditions, but a GUESS being -45.0.

It was suggested that the computer program written to generate the solutions and plots could be run up to a point just prior to the breakdown, and then using the final values as a new set of initial conditions. Unfortunately, this helped very little. As a check as to whether the computer program was at fault, a test program was run on a Hewlett-Packard HP-67 hand calculator using the same conditions, and it was found that the same situation occurred with essentially the same outcome. This was double-checked by running a Runge-Kutta routine designed for the HP-67 by Hewlett-Packard [59], but with the same result.

At present, the question as to what occurred, and why, remains unanswered. However, it would appear from the plots obtained for the region of $q \leq 20.0$ that no problems would be experienced for a positive value of the characteristic number. Possibly, a small negative value might do so, but this was not investigated. There are indications that this may even extend up to $q = 30.0$. Problems quite definitely develop for $q = 40.0$, and it appears that regardless of what the value for the characteristic number is, deviations from the series approximation will occur.

There does not appear to be any symmetry in the severity of the deviations as far as initial conditions are concerned. An example of

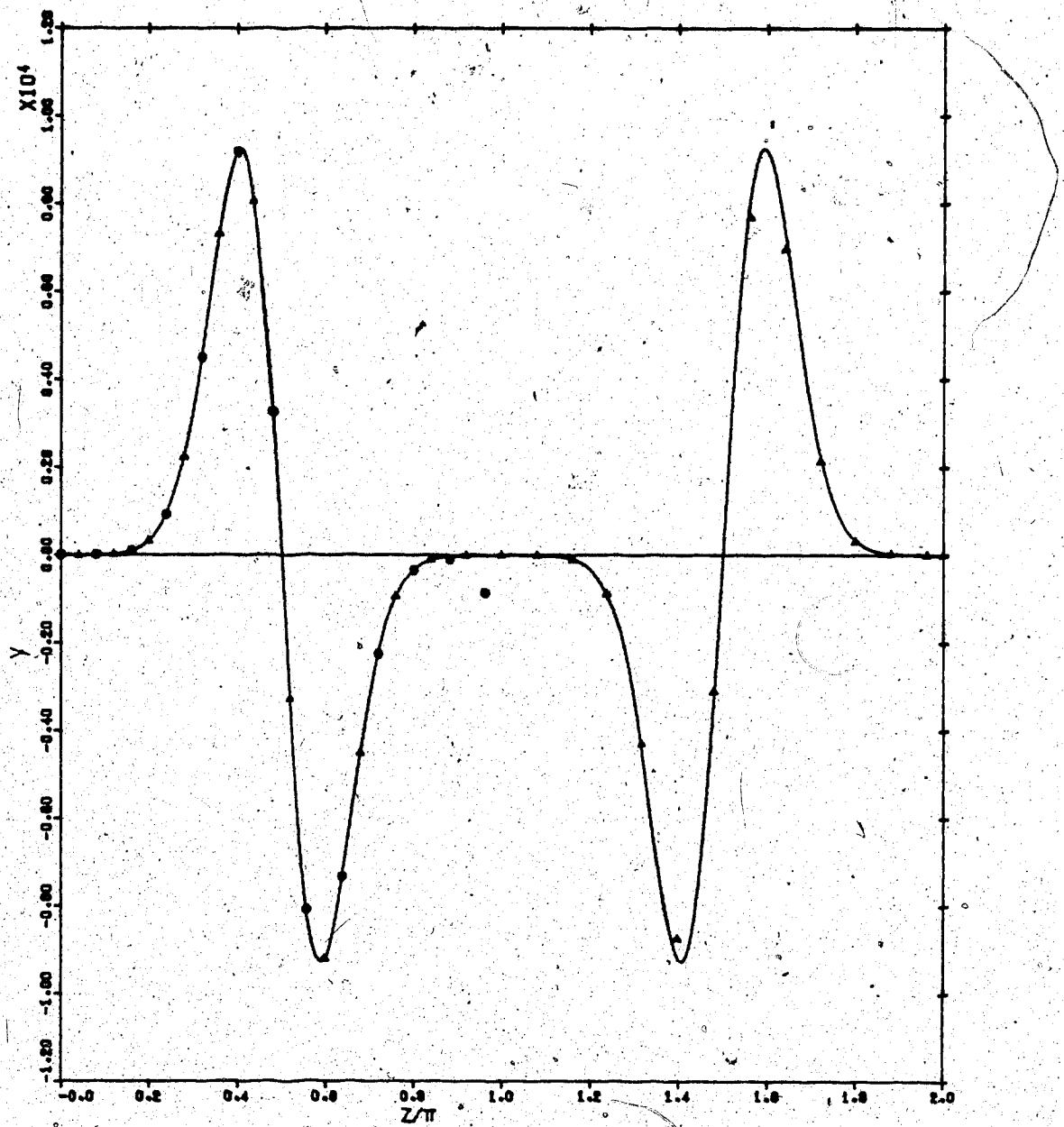
FIGURE 6.14 - SOLUTIONS TO MATHIEU EQUATION FOR

$a_{\text{GUESS}} = -50.0$, $q = 40.0$, $\Delta z = \pi / 400.00$,
 $y(0) = 1.0$, $y'(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5].)

CHAR. NUM. (EST. VALUE) = -50.0000
CHAR. NUM. (FINAL VALUE) = -43.3522
 $q = 40.0000 \quad \Delta z = \pi/400.0$
 $y(0) = 1.0000 \quad y'(0) = 0.0000$

— SERIES APPROXIMATION
○ PIECEWISE LINEARIZATION
△ RUNGE-KUTTA ▲



this is the case of a GUESS = -50.0 and q = 10.0. For the conditions of $y(0) = 1.0$, and $y'(0) = 0.0$, there is a slight deviation near the end of the first cycle, but only a fraction of the maximum displacement. However, for the same characteristic number estimate and q, but $y(0) = 0.0$ and $y'(0) = 1.0$, a totally different situation arises in which the maximum displacement for the piecewise linearization ~~exceeds~~ the other solutions. Though this is an isolated example, there may possibly be a problem here.

As had been shown in the previous chapters, the program had been stripped to its essential calculations, removing comment and output statements, and the timings of the three solutions were obtained [27] for selected cases. The results of this can be seen in Figures 6.15 - 6.17. This was run on the Amdahl 470V/8 [26].

For the piecewise linearization and the series approximation, a linear relationship with the number of divisions of π . (and hence, the size of the increment of z) and the execution time exists, but the Runge-Kutta method [5] displays some unusual features. First, its execution times are almost always larger than those for the other two solutions, except for Figure 6.15 and the smaller values of Δz in Figure 6.16. In fact, it is considerably larger than what was obtained for the piecewise method, being larger by about a factor of 10 for a large Δz but down to four times as large for the smaller values.

The other unusual feature is the fact that the Runge-Kutta results [5] show some scatter, particularly seen on Figure 6.16. This is different than what was observed for the hard spring equations (in which the results were linear) and for the Van der Pol equation where the results were exponential.

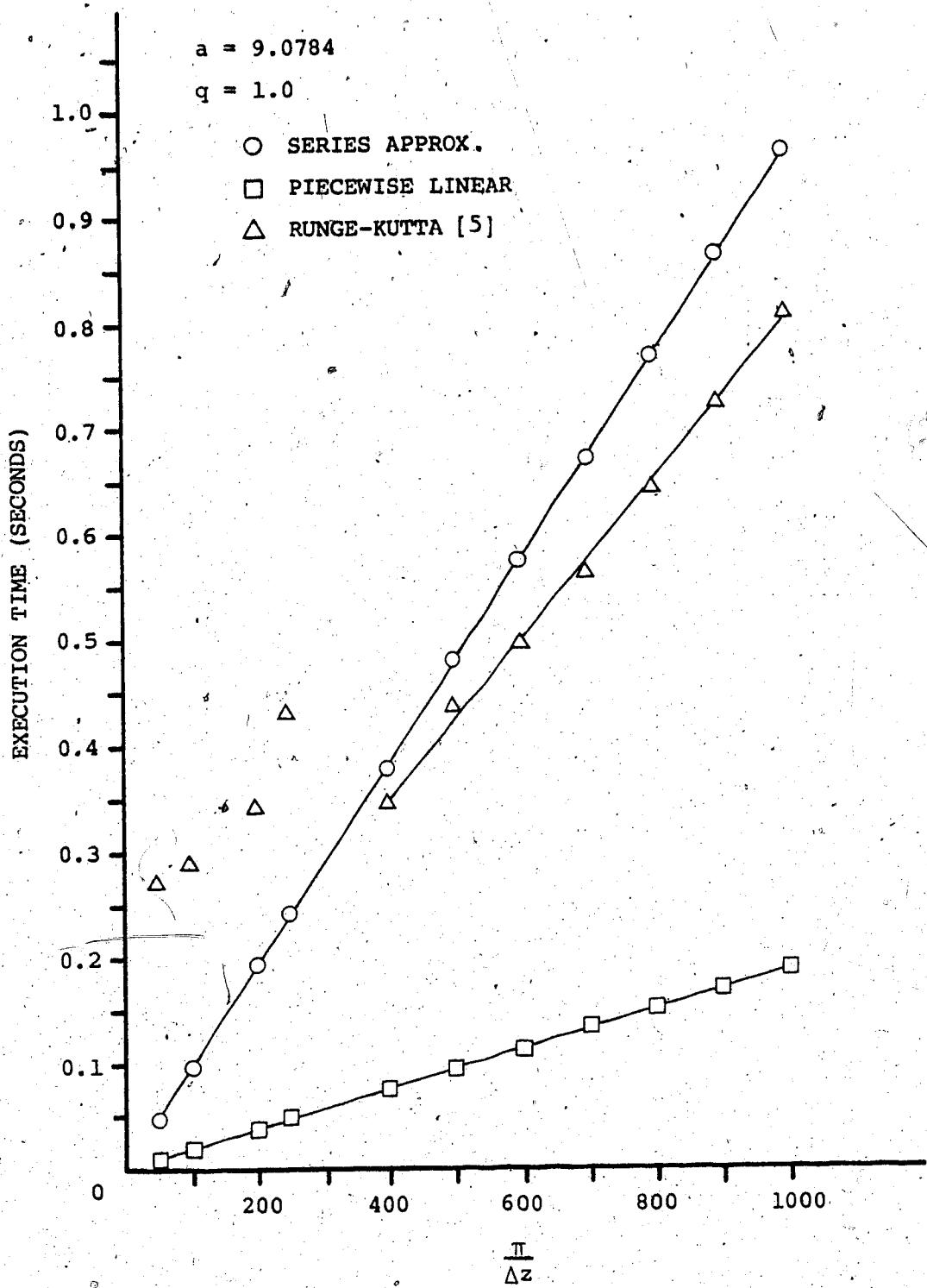


FIGURE 6.15 - EXECUTION TIMES FOR SOLUTIONS TO
 MATHIEU EQUATION FOR $a_{GUESS} = 10.0$, $q = 1.0$,
 $y(0) = 1.0$, $y'(0) = 0.0$ [27]

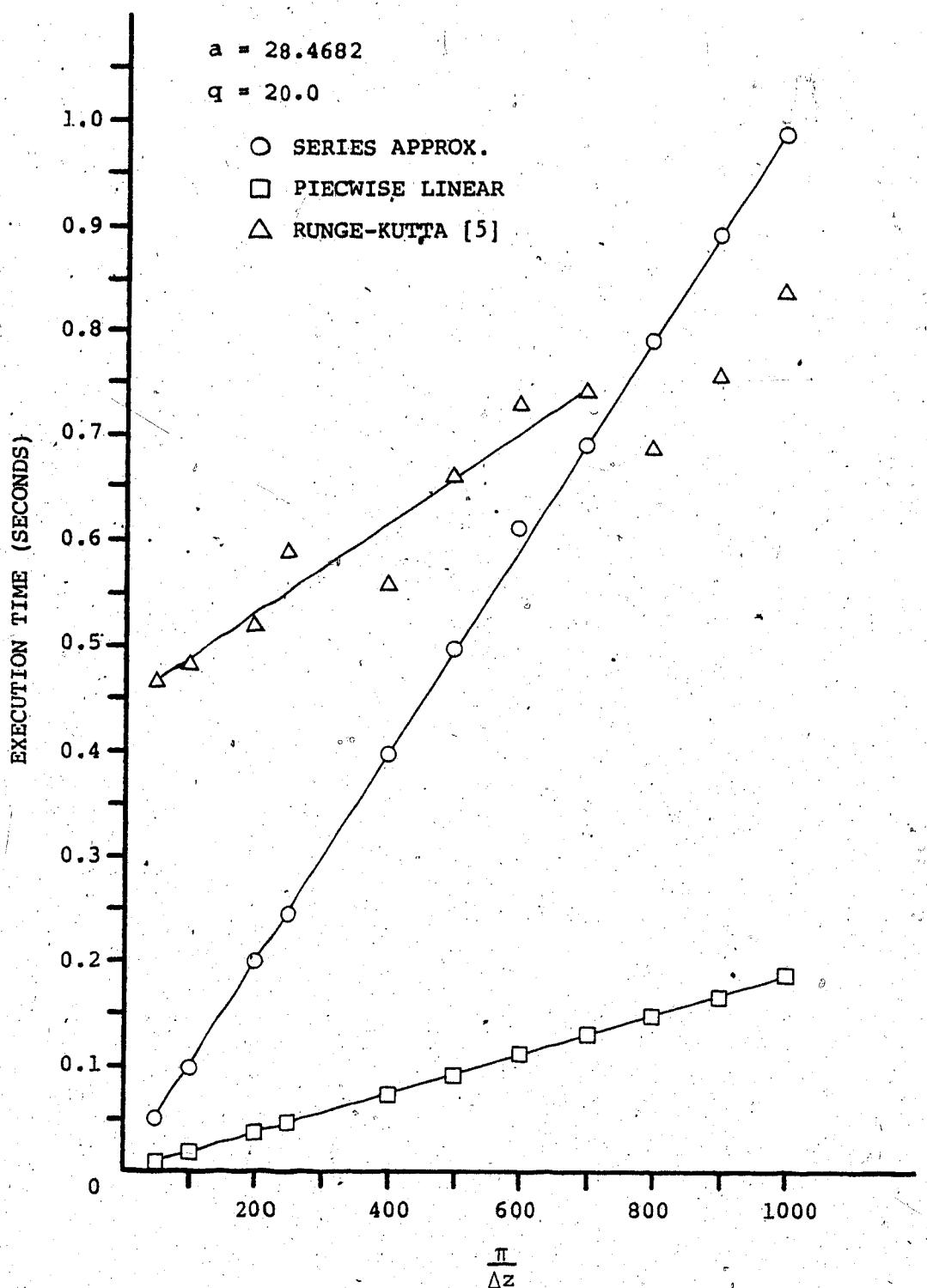


FIGURE 6.16 - EXECUTION TIMES FOR SOLUTIONS TO MATHIEU EQUATION FOR $a_{GUESS} = 30.0$, $q = 20.0$, $y(0) = 1.0$, $y'(0) = 0.0$ [27]

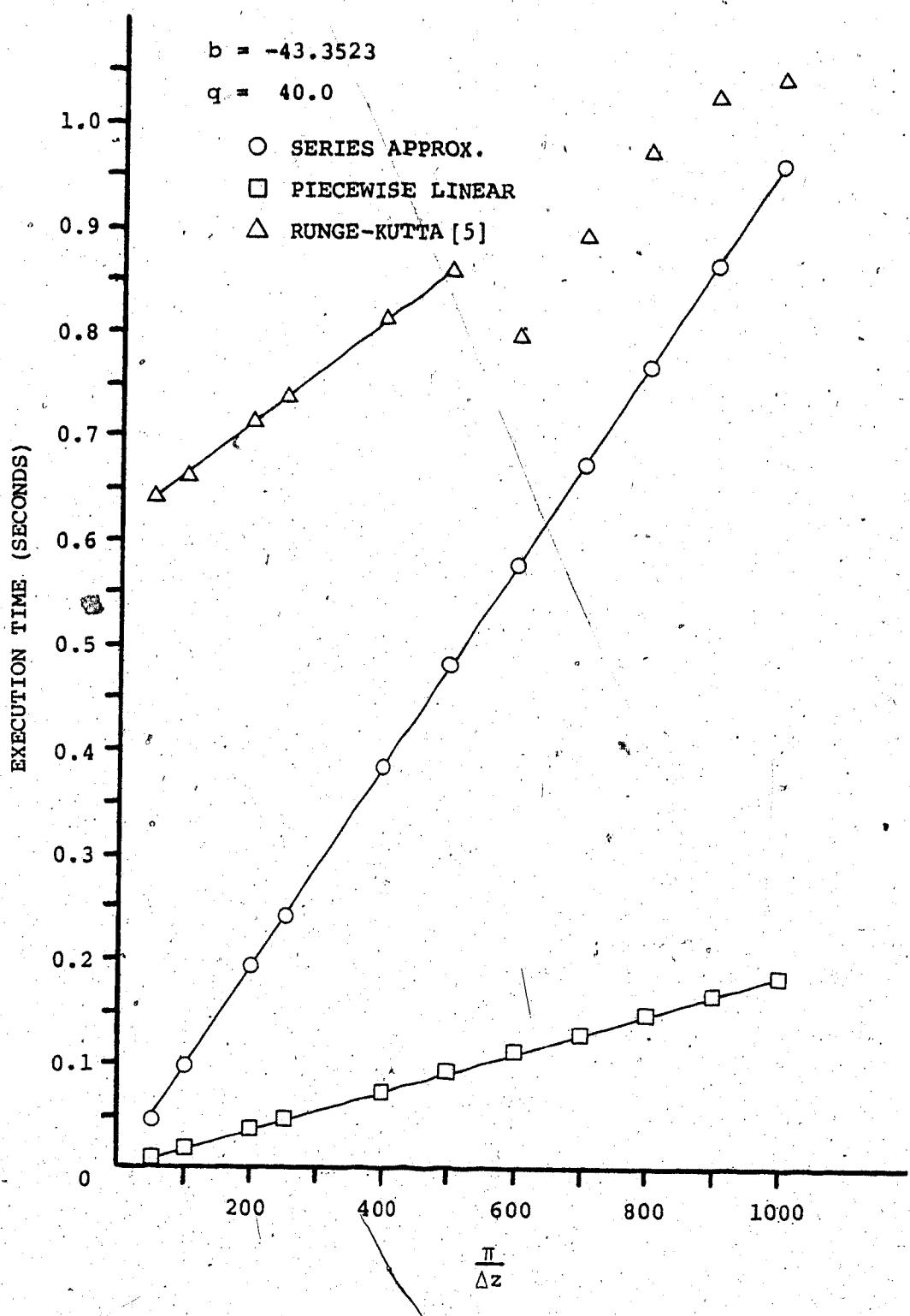


FIGURE 6.17 - EXECUTION TIMES FOR SOLUTIONS TO
MATHIEU EQUATION FOR $a_{GUESS} = -50.0$, $q = 40.0$
 $y(0) = 1.0$, $y'(0) = 0.0$ [27]

This behaviour was discussed with a consultant from the University of Alberta Computing Services [60] in an attempt to determine what had brought this about. A suggested explanation is that the routine that generated the Runge-Kutta results [5] may have been affected in whole or in part by the actual size of the Δz used. However, the exact cause of the execution time variation remains unknown.

VII. DISCUSSION

As was mentioned in Chapter 2, the method of tangents for solving the undamped hard spring equation had a greater execution time than the method of chords because of the extra calculations that had to be carried out in order to arrive at a solution. The period of oscillation as calculated by the former method was closer to the value obtained by the approximation to the exact solution [4], [15], [17] than the period calculated by the latter one, but the method of chords had a closer fit to the solution curve of this approximation than did the one using tangents.

For 5 segments, the period as calculated by the approximation to the exact solution [4], [15], [17] for $a = 1.0$ and $b = 2.0$ is 4.004308722, while the one obtained by chords is 3.976017631 with the one by tangents, 4.018171439. The period found by the method of chords was 0.707% less than (2.2.5) [17], while the value for the method of tangents was 0.346% larger. If one were to consider the case of 100 segments for $a = 1.0$ and $b = 0.15$, where (2.2.5) [17] was found to be 5.958299670, the chords gave 5.958281464 (0.0003% less) while the tangents gave 5.958310457 (0.0002% greater). So, the tangents gave a more accurate calculation, as far as the magnitude of error is concerned, for the period.

However, in judging a method of solution, the period itself is not necessarily the only standard. The accuracy of the fit of the calculated solution with respect to the one that had been chosen to be the benchmark [4], [15], [17] is perhaps more important, and it is here that the method of tangents starts showing a small, but significant,

difference. If one were to look at Figure 2.8, one can see that the tangents are off by a slight amount, while the chords are much closer. For Figures 2.6 and 2.7, this does not become immediately apparent, which indicates that the values of a and b would have some bearing on the closeness of the fit.

As mentioned before, it was because the method of chords gave a closer fit that this method was adopted throughout the remainder of the thesis as the method of piecewise linearization, as well as the fact that it was quicker to calculate.

The execution time provided some interesting results. Since the solution chosen to be the standard was based on a packaged program library [5], one would have expected it to have a shorter execution time than the other solutions, but since (2.2.4) [15] was calculated each time, this apparently unusual result appears to be resolved. It should be noted that while setting up the timing program, the author had run it in which (2.2.4) [15] was calculated only once and the solutions generated after that, and the approximation to the exact solution did, in fact, turn out to be faster by about 10% than that of the method of chords for most of the runs.

The two cases of the damped hard spring were an extension of the undamped one. For the situations examined, a good agreement between the results of the piecewise solution and those of the Runge-Kutta method [5] was obtained, which leads one to conclude that for the size of the displacement used, the piecewise linearization was a suitable approximation to the chosen standard, since an exact solution is not available.

Because the same calculations are carried out for each iteration, the timing results (using [27]) were linear with respect to the number

of divisions per first quarter-cycle will occur, and also, the quantities being incremented were different in each solution. Also, as expected, the Runge-Kutta method [5] used less time than the piecewise method. This can be seen if one considers that this is from a packaged program library, and so, should have the advantage of development and modifications. In fact, the IMSL package used was in its seventh edition [5], and presumably, should be as efficient as possible, as compared to the piecewise linearization portion of the computer program used to generate the solutions.

What is unusual is that for a large increment (that is, a small number of divisions per first quarter-cycle), the timings show a large deviation, which probably means that the increment chosen was so large that, in fact, the solution does not have time to converge (if a time was obtained) or that both solutions have to work harder to achieve a result. This appears further on for the Van der Pol equation.

The hard spring equation piecewise linear solutions gave a close correspondence with the results from the Runge-Kutta method [5]. However, this was not the case for the piecewise linearization of the Van der Pol equation and the same Runge-Kutta method [5], and for the piecewise linear results for the Mathieu equation and the approximation to the exact solution.

The apparent "phase difference" noticed in the Van der Pol equation results, and which apparently was resolved by reducing the size of the time increment used, was the first sign that the piecewise linearization could have some limitations. Some insight was gained by considering the subject of stability, since this may have some bearing on the results obtained for the Mathieu equation.

Several things should be noted first before continuing. One is that because of the apparently strange results obtained for the Van der Pol equation for the higher values of μ , there may be a limit beyond which the piecewise linearization becomes unreliable. In other words, for the equation parameters and increment sizes, there may be an area in which this method is ineffective. This is indicated by Figures 5.21 and 5.22 [38], [39].

If this is the case, then, can one predict if, and when, this will occur for a particular situation? Is it actually due to the numerical instability or possibly machine error? The answer to the first question is that it may not always be possible to do so. This is indicated by Greenspan [61], and it would therefore appear that a trial-and-error approach is the only sure method of determining if an equation becomes unstable for a given set of parameters.

The question now is whether this error was caused by instability or machine error. The internal tolerance for calculations like the Newton-Raphson [30] relation that was used for determining the remaining time until a peak or trough had been reached had been set to smaller values to ensure that perhaps this may not have been the source. This was because the solutions appeared to be all right for the smaller values for μ . Also, the tolerance for the Runge-Kutta program had also been reset to a smaller value [5], but the result was that virtually no change was noticed in this solution. Since the smaller value for the time increment appeared to resolve the problem, it was concluded that numerical instability may have caused it.

This led to the consideration of the accuracy of the solution to (5.1.1) obtained by using the Runge-Kutta method [5]. Figures 5.21 and

5.2.2 [38], [39] indicated that it may become inaccurate with a large time interval, but perhaps not to the same extent as the piecewise linearization. It can therefore be concluded that the results from the Runge-Kutta method [5] may be misleading if factors like this are not taken into account.

Next, the timing runs were done [27]. The piecewise linearization had a parabolic relationship between execution time and the size of the time increment, and the Runge-Kutta [5] had an exponential relation. This can be easily explained by referring for a moment back to the results for the damped hard spring equations.

There, for a small number of divisions for the first quarter-cycle, the times for both solutions deviated significantly from the remainder of the results, which indicated that both solutions had to go through more iterations at this stage. The same probably occurs here, since there were some runs for which times were not obtained because the computer kept exceeding its set time limit [31]. This occurred for the higher values of Δt_p and Δt_{RK} . For a small value of Δt a large number of iterations had to be carried out, though, because of the small increment, convergence would be rather quick [60].

The aspect of instability occurs once again with the Mathieu equation, and this will now be examined. For the lower values of q , there was generally no problem with the characteristic number was large and negative, and what makes it particularly puzzling is the fact that the situation described by Figure 6.7 does not seem to follow any particular pattern relating to the other solutions. For the Van der Pol equation, the deviations at least looked like the Runge-Kutta solution [5], but expanded along the time axis. However, for the Mathieu

equation, this does not seem to happen.

Figure 6.7 may very well represent part of the unstable region, as had just been discussed. Figures 6.9, 6.11, 6.13, and 6.14 were interesting in the fact that the maximum amplitude of the piecewise linearization completely dwarfed the other solutions as was seen when the initial results were obtained. (It should be noted that the plots were scaled by taking the largest amplitude of the three solutions, and so, this result will occur.) The program was temporarily modified to show the other solutions, and so, the piecewise linearization results end up going off scale.

In particular, Figures 6.13 and 6.14 display that changing the size of Δz does not seem to have any major effect on this phenomenon. The analysis is further complicated through the piecewise linear results following the series approximation to the exact solution quite closely until nearly the end of the first half-cycle, and then becoming unstable.

Though not shown here, it should be noted that the results from the Runge-Kutta method [5] had exhibited similar behaviour. This was observed by the author when developing the program for solving the Mathieu equation for the same initial conditions for Figures 6.13 and 6.14, but using an a_{GUESS} of -45.0 instead of -50.0. It became unstable to a smaller extent than the piecewise linearization and took place in roughly the same area, while the series approximation was "well-behaved".

Perhaps some insight can be gained by examining the subject of numerical instability. First, one must consider what sort of instabilities exist.

Most texts on numerical analysis will give four definitions, depending upon what the situation in question is, such as [62], [63], [64], [65], [66]. (It should be noted that the definitions to be given are as used in the references.)

"Inherent" instability occurs when a small change in the starting value will cause large changes in the solution to occur, but this is independent of the numerical method chosen to solve the problem [62]. It is the equation itself that is insensitive, and not the means used to solve it [63].

"Partial" instability arises when the solution deviates significantly from the true one as more steps are taken. The situation is dependent upon the equation that is being solved, the method used, and especially the step size. This appears to be resolved when a smaller increment is chosen [64].

"Strong" instability arises when one solves the equation, but additional terms arise that do not vanish as the step size gets very small. These terms increase faster than the desired solution, and lack of convergence as well as stability is implied here [65], [66].

"Weak" instability is defined as having the same behaviour as what characterizes strong instability occurs, but the solution converges [65].

In view of these definitions, the problems with the Mathieu equation piecewise linear solution seem to display a combination of all of these, but cannot be completely described by any one of them. As was mentioned before, the initial conditions were changed by a slight amount, but this did not give rise to the radical changes that "inherent" instability would imply. "Partial" instability [64] seems to

describe what had occurred with the Van der Pol equation, but still, a change in the step size did not seem to matter much, though there is some indication that if one were to take an extremely small step size (such as on the order of $\pi/5000$), a partial success is achieved in that the amplitude of the piecewise linear method is reduced by at least a factor of 10, but the "takeoff" point for this irregularity does not seem to have been affected much by being delayed for a few more increments, as compared to what happened with the Van der Pol equation. So, it would seem that "partial" instability [64] can be eliminated as a probable cause.

After breakdown, the solution appears to follow a pattern in the resulting curve rather than having the points randomly distributed. This may be an indication of "weak" instability [65], but because there was a good correspondence between the piecewise linearization and the series approximation to the exact solution, it cannot be the sole cause. The possible clue concerning "weak" instability [65] was obtained from the original plot from which Figure 6.9 was derived (Figure 7.1)

Since no single definition of instability fully describes the observed results, it appears that the reasons for the failure of the piecewise linear method are perhaps more complex than first imagined.

Figure 7.2 is a plot of the characteristic numbers obtained versus q , and the distinction is made between those values for which serious deviations occur and those for which the piecewise linear solution is reasonably well-behaved. There appears to be a transition zone between which the solutions go from stable to unstable. Unfortunately, there is not enough data available to give a better resolution of the boundaries, and so, Figure 7.2 has to be taken as a very rough estimate. What it

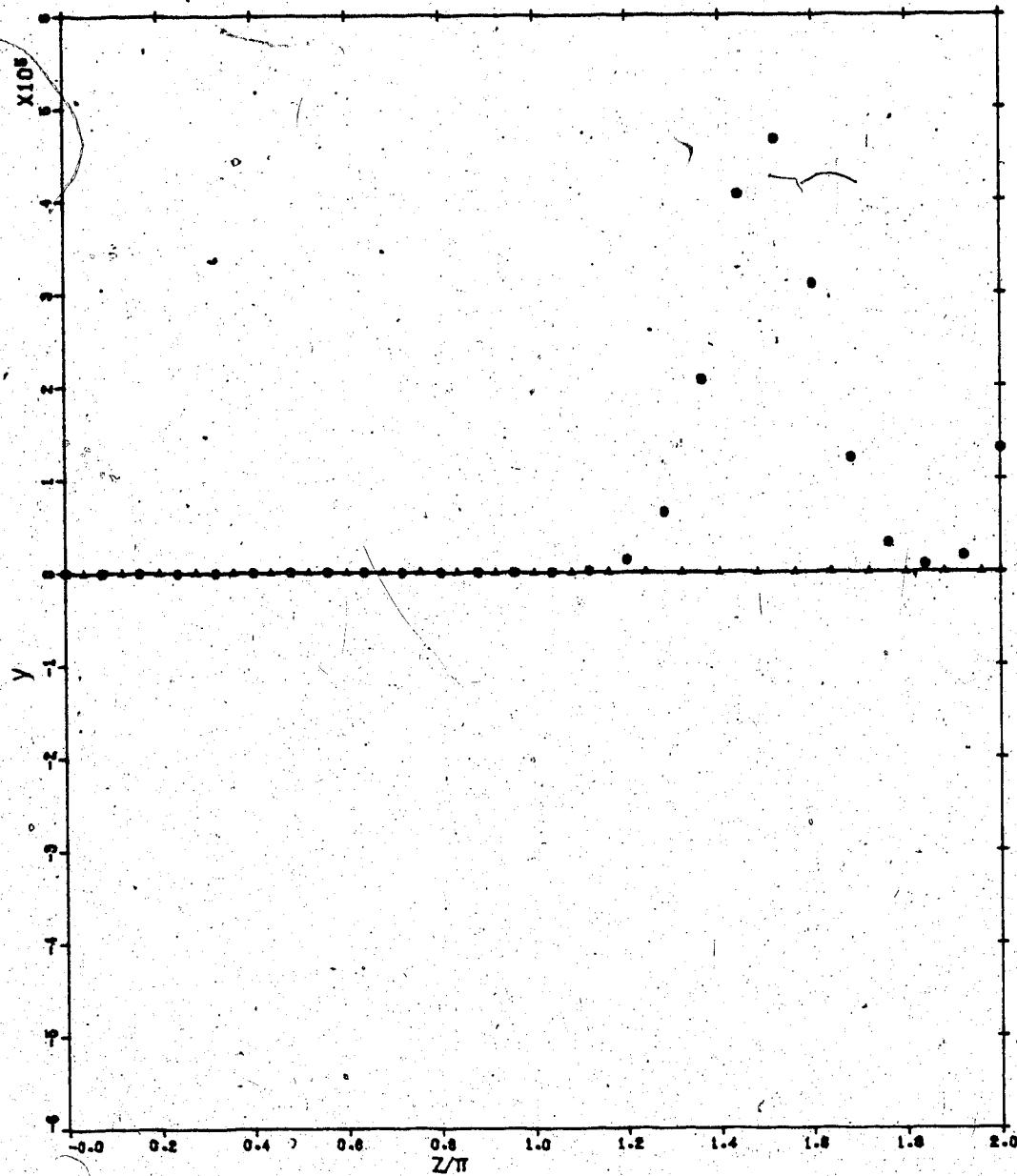
FIGURE 7.1 ♦ SOLUTIONS TO MATHIEU EQUATION FOR

$a_{\text{GUESS}} = -50.0$, $q = 20.0$, $\Delta z = \pi/200.00$,
 $y(0) = 0.0$, $y'(0) = 1.0$

(N.B.t Runge-Kutta solution uses [5].)

CHAR. NUM. (EST. VALUE) = -50.0000
CHAR. NUM. (FINAL VALUE) = -31.3134
 $q = 20.0000 \quad \Delta z = \pi/200.0$
 $y(0) = 0.0000 \quad y'(0) = 1.0000$

— SERIES APPROXIMATION
○ PIECEWISE LINEARIZATION
△ RUNGE-KUTTA



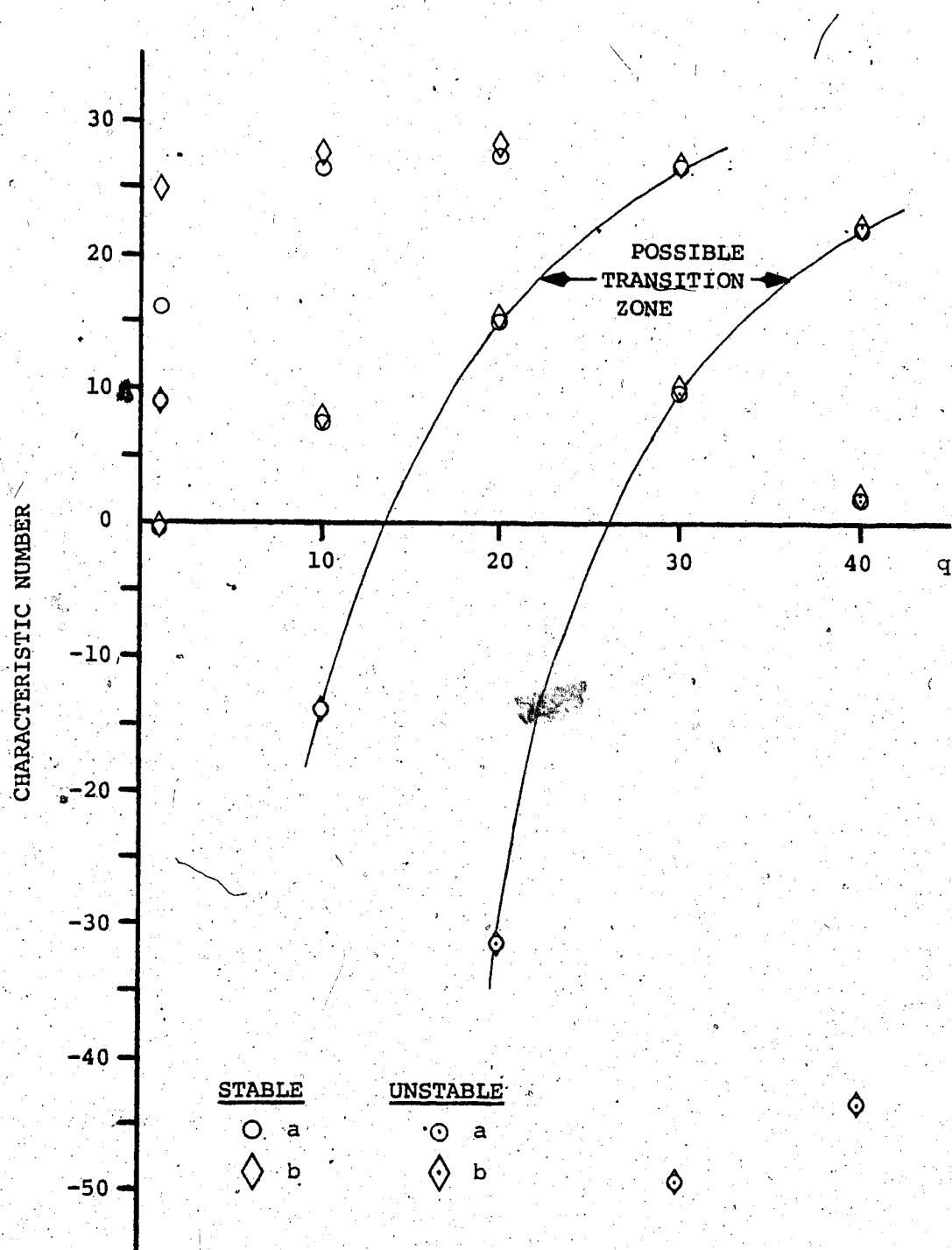


FIGURE 7.2 - STABILITY CHART FOR PIECEWISE LINEAR SOLUTION
TO MATHIEU EQUATION

does indicate is that if this problem of the runaway piecewise linear solution cannot be resolved, this chart (or rather, a more accurate version of it) would give one an idea as to where the piecewise linear solution to the Mathieu equation becomes ineffective and another method should be pursued.

The timing results really posed no major problems as far as most of the values were concerned [27]. The piecewise linear method was fairly straightforward, and so, one can expect a linear solution, since as the number of increments is increased, the same calculations have to be done that many times more. The same applies to the series approximation to the exact solution, where the solution is mainly composed of the manipulation of the coefficient ratios.

The Runge-Kutta method [5] did give some unusual results. First, the magnitude of its times were suspect since one would expect the method to be faster than the others based on results for (3.1.1), (4.1.1), and (5.1.1). However, since the Mathieu equation is actually a linear one with a time-varying coefficient, it may be a difficult one for the Runge-Kutta routine [5] to manipulate. The piecewise linear method merely approximates the coefficient with a constant, and the series approximation to the exact solution is based upon the Mathieu functions.

The scatter in the timing results may have been a consequence of the size of the interval of z rather than the equation [60]. It appears that DVERK [5] itself may have been directly affected by the magnitude of Δz since (6.1.1) is a different type of equation than had been encountered in the previous chapters. How it is affected by it, and subsequent steps it executes could also explain why the execution times,

as determined by use of [27], were so large compared with those for the others.

VIII. CONCLUSIONS

1. The accuracy of the piecewise linear solution with respect to the one being compared will vary with the equation being examined. This is dependent upon the presence of damping, the degree of nonlinearity (if any) and whether the coefficients in these equations are constant or variable.
2. For those cases in which correspondence between the piecewise linear solution and the solution being compared is poor, there appear to be areas for which the piecewise linear method is unstable, or at least limited. The degree to which this occurs will depend upon the equation parameters and constants, as well as the size of the increments chosen, whether displacement or time.
3. The type of instabilities displayed by the piecewise linear solutions may be equation-dependent. That is, the instability seen in the Van der Pol equation appeared to be different than that displayed by the Mathieu equation. This may have to be confirmed by further work with these two equations plus the investigation of others.
4. For large increments of time or displacement (depending upon which equation was being solved), the piecewise linear method can become uneconomical since it will have a larger execution time compared with a smaller increment, and quite often can exceed the time limit set for the program used. However, under certain circumstances, the execution time of the piecewise linear method can be close to that of the chosen standard solution (such as the Runge-Kutta method [5] used as a comparison for the Van der Pol equation), or even cheaper (as was seen for the undamped hard spring and Mathieu equation).

5. Generally, the piecewise linearization is slower than the solution being compared for the system being examined except in the case of the undamped hard spring and the Mathieu equations. For the former, it was because the approximation to the complete elliptic integral of the first kind [15] had to be calculated each time the number of segments was increased, while for the Mathieu equation it was because the piecewise linearization was simpler than the approximation to the exact solution and the Runge-Kutta method [5]. It can be seen here, then, that the method of approximating the equation will determine the speed (and hence, the cost) of a run.

6. The tangential method of linearization is slower than that using chords, because of the various calculations that have to be carried out. The correspondence with the approximation to the exact solution [4], [15], [17] appears to be better for chords than for tangents, though the latter is more accurate when it comes to calculating period for the undamped hard spring equation.

IX. RECOMMENDATIONS FOR FURTHER WORK

Several questions arising from the results obtained still need to be answered. At the same time, there are some areas that can be investigated in more detail, extending past what has already been examined. The following, then, are some suggested subjects for further study.

A. Mathieu's Equation

The first thing that comes to mind is the situation with the instability of the piecewise linear method. The exact nature as to what was occurring has to be resolved, since it is unusual that the piecewise linearization solution had behaved rather well until it apparently broke down. The actual mechanism behind this has to be examined in depth to determine not only what is taking place, but also why, and if there is any means of rectifying this.

On the other hand, suppose that the reason for the instability observed in Figures 6.7, 6.9, 6.11, 6.13*, 6.14, and 7.1 cannot be resolved. One alternative is to try and pursue other methods of solving this equation, using a different means of approximating the linear segment. If, however, all attempts to solve this problem fail, the only other thing that can be suggested is to obtain more data and fill in the gaps in Figure 7.2, to see if a boundary between stability and instability can be clearly defined.

In McLachlan [67], reference was made to an earlier paper of his [68] that dealt with fractional orders for the solutions. This reference, in turn, referred to a paper by Ince on the same subject [69].

In [68], a method for determining the order of the solution for a

particular a GUESS and q is given, based on [69]. This may make the solution of (6.1.1) more general in nature compared with what had been done in Chapter VI, as fractional orders can now be included.

B. Two-Degree-of-Freedom Nonlinear Equations

Since many physical phenomena can be described by two-degree-of-freedom nonlinear equations, one possible approximate method of solution is piecewise linearization. However, the major difficulty is in the separation of the coupled equations that describe the situation into separate modes.

Some work on this separation has already been done in this area by Rosenberg [70], [71], which deals with specific classes of two-degree-of-freedom nonlinear differential equations. He also produced a major work dealing with this subject, and it appears that this may be of a more general nature [72].

Since this subject was merely touched upon, some work would be required in seeing what is needed for solving specific cases. Once the equations have been uncoupled (if such is possible for the particular situation being examined), the piecewise linear method can then be applied. These references may provide a starting point.

REFERENCES

1. Cunningham, W. J. Introduction to Nonlinear Analysis. New York, McGraw-Hill Book Company, 1958, p. 135-140.
2. Ibid., p. 133-135.
3. Ibid., p. 123-133.
4. Madderom, P. "JCBYF" in UBC Function--Mathematical (or Special) Functions. Vancouver, B. C., University of British Columbia Computing Centre, June, 1979, p. 40-41.
5. "DVERK", in IMSL Library Reference Manual, Seventh Edition, Vol. 1. Houston, Tex., IMSL, Inc., Jan. 1979, DVERK1-DVERK9.
6. Cunningham, op. cit., p. 66.
7. This is based on a diagram found on p.25, ch. 2 of the course notes for ELEC 557 given from Sept., 1979 to Apr., 1980 by Dr. A. C. Soudack, Professor, Dept. of Electrical Engineering, University of British Columbia, Vancouver, B. C.
8. Cunningham, op. cit., p. 75.
9. McLachlan, N. W. Ordinary Non-Linear Differential Equations In Engineering and Physical Sciences. London, Oxford University Press, 1956, p. 24-27.
10. Cunningham, op. cit., p. 74.
11. Soudack, op. cit., p. 5-63 - 5-86.
12. Cunningham, op. cit., p. 77-78.
13. Soudack, op. cit., p. 5-84 - 5-85.
14. Milne-Thomson, L. M. "Elliptic Functions", in Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, ed. M. Abramowitz and I. G. Stegun. New York, Dover

- Publications, Inc., 1972, p. 590.
15. Ibid., p. 591.
 16. Milne-Thomson, L. M. "Jacobian Elliptic Functions and Theta Functions", in Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, ed. M. Abramowitz and I. G. Stegun. New York, Dover Publications, Inc., 1972, p. 571.
 17. McLachlan, op. cit., p. 27.
 18. Timoshenko, S., et. al. Vibration Problems in Engineering, Fourth Edition. New York, John Wiley & Sons. Inc., 1974, p. 190-193.
 19. Ibid., p. 191.
 20. Riddle, Douglas F. Calculus and Analytic Geometry. Belmont, Calif., Wadsworth Publishing Co., Inc., 1970, p. 6.
 21. Timoshenko, et. al., op. cit., p. 192.
 22. Franklin, Phillip. "Algebra", in Mark's Standard Handbook for Mechanical Engineering, Eighth Edition, ed. Theodore Baumeister, et. al. New York, McGraw-Hill Book Company, 1978, p. 2-19.
 23. C R C Standard Mathematical Tables, 25th Edition, ed. William H. Beyer. Boca Raton, Florida, C R C Press, Inc., 1979, p. 170.
 24. Riddle, op. cit., p. 21, 22.
 25. University of Alberta Computing Services Overview 0120-0480, Computing Services. Edmonton, Alta., April 1980, p. 8.
 26. University of Alberta Computing Services Bulletin, Edmonton, Alta., N1S11.0681, June 5, 1981, p. 1.
 27. University of Alberta Computing Services subroutine TIME. Its use was suggested by fellow graduate student, Roger Toogood, Dept. of Mechanical Engineering.
 28. Based on a diagram in Soudack, op. cit., p. 48, ch. 3.

29. Timoshenko, et. al., op. cit., p. 63-66.
30. Greenspan, Donald. Discrete Numerical Methods in Physics and Engineering. New York, Academic Press, Inc., 1974, p. 15.
31. This time limit was set by the author through his on-campus terminal which was connected to the University of Alberta Computing Services facilities.
32. Soudack, op. cit., p. 69, 70, ch. 3.
33. Timoshenko, et. al., op. cit., p. 64.
34. Timoshenko, et. al., op. cit., p. 63-71.
35. Timoshenko, et. al., op. cit., p. 66.
36. Timoshenko, et. al., op. cit., p. 71.
37. Timoshenko, et. al., op. cit., p. 69.
38. Based on diagrams in Soudack, op. cit., p. 72, ch. 3.
39. Based on diagrams in Soudack, op. cit., p. 71, ch. 3.
40. University of Alberta Computing Services Reference R131.0280, CGPL/CGPL2. Edmonton, Alta., February 1980, p. 1-12.
41. McLachlan, N. W. Theory and Application of Mathieu Functions. Oxford, Clarendon Press, 1947, p. 10.
42. Ince, E. L. Tables of the Elliptic Cylinder Functions, Proc. Roy. Soc. Edinburgh, Vol. 52, 1932, p. 355-423.
43. McLachlan, Theory and Application of Mathieu Functions, p. 10-41.
44. Ibid., p. 18.
45. Ibid., p. 21.
46. Ibid., p. 28.
47. Ince, op. cit., p. 358.
48. Ince, op. cit., p. 359.
49. Ince, op. cit., p. 360.

50. Ince, op. cit., p. 363.
51. McLachlan, Theory and Application of Mathieu Functions, p. 37.
52. Ibid., p. 29-31.
53. Ibid., p. 31.
54. Ibid., p. 30.
55. Ibid., p. 37.
56. Ibid., p. 25.
57. Ince, op. cit. p. 364-375.
58. Timoshenko, et. al., op. cit., p. 3, 4.
59. Program MAI-11A, "Differential Equations", Hewlett-Packard HP-67/HP-97 Math Pac 1, Corvallis, Oregon, 1976, p. 11-01 - 11-04, L11-01 - L11-02.
60. This information was obtained through consultation with Dr. R. Torgerson, Senior Analyst, University of Alberta Computing Centre, Edmonton, Alta.
61. Greenspan, op. cit., p. 52.
62. Fröberg, Carl-Erik. Introduction to Numerical Analysis, Second Edition. Reading, Mass., Addison-Wesley Publishing Co., Inc., 1969, p. 277.
63. Dorn, William S. and McCracken, Daniel D. Numerical Methods with Fortran IV Case Studies. New York, John Wiley and Sons, Inc., 1972, p. 389.
64. Ibid., p. 378-379.
65. Mayers, D. F. "Stability of Step-by-Step Methods", in Numerical Solutions of Ordinary and Partial Differential Equations, ed. L. Fox. Reading, Mass., Addison-Wesley Publishing Co., Inc., 1962, p. 48.

66. Ibid., p. 51.
67. McLachlan, Theory and Application of Mathieu Functions, p. 79, 378.
68. McLachlan, N. W. "Computation of the Solutions of Mathieu's Equation", Philosophical Magazine, Vol. 36, Series 7, 1945, p. 403-414.
69. Ince, E. L. "Mathieu Functions of Stable Type", Philosophical Magazine, Vol. 6, Series 7, 1928, p. 547-558.
70. Rosenberg, R. M. "On Normal Vibrations of a General Class of Nonlinear Dual-Mode Systems", Journal of Applied Mechanics, Transactions of the ASME, Vol. 83, Series E, June, 1961, p. 275-283.
71. Rosenberg, R. M. "The Normal Modes of Nonlinear n-Degree-of-Freedom Systems", Journal of Applied Mechanics, Transactions of the ASME, Vol. 84, Series E, March, 1962, p. 7-14.
72. Rosenberg, R. M. "On Nonlinear Vibrations of Systems with Many Degrees of Freedom", Advances in Applied Mechanics, Vol. 9. New York, Academic Press, p. 155-242.
73. Buttuls, Peter and Dembo, Stevyn. University of Alberta Computing Services Tutorial T11.1277, Digital Plotting System. Edmonton, Alberta, University of Alberta Computing Services, December, 1977.
74. "Mathematical Notation - Special Functions", in McGraw-Hill Dictionary of Scientific and Technical Terms, Second Edition, ed. Daniel N. Lapedes. New York, McGraw-Hill Book Company, 1978, p. A9.
75. Cress, Paul, et. al. FORTRAN IV with WATFOR and WATFIV. Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1970.

APPENDIX

The following programs were used in obtaining the results for this thesis:

- A. HSPRING - Hard spring equation (undamped)
- B. HSPRINGC - Hard spring equation (linear damping)
- C. HSPRINGQ - Hard spring equation (nonlinear damping)
- D. VANDERPOL - Van der Pol equation
- E. MATHIEU - Mathieu equation.

These programs were developed and run on the facilities of the University of Alberta Computing Services [25], [26], with the plot routines used being [40], [73]. Material upon which these programs are based is found in:

HSPRING - [4], [8], [9], [10], [11], [12], [14], [15],
[16], [18], [20], [22], [23], [24];
HSPRINGC - [5], [18], [20], [29], [30];
HSPRINGQ - [5], [18], [20], [29], [30], [74];
VANDERPOL - [5], [30], [32], [34], [38], [39], [74];
MATHIEU - [5], [34], [42], [43], [58].

The signum function as defined in [74] was used in HSPRINGQ and VANDERPOL. The major reference for the program code was [75].

```

1 C#####
2 C#
3 C# PROGRAM HSPRING
4 C#
5 C#
6 C#
7 C# THE FOLLOWING PROGRAM CALCULATES AND COMPARES THE SOLUTIONS
8 C# FOR THE EQUATION:
9 C#
10 C# X(DOUBLE-DOT)+A*X+B*(X+3)=0
11 C#
12 C# OBTAINED BY PIECEWISE LINEARIZATION (USING CHORDS AND
13 C# TANGENTS) WITH THAT OF AN EXACT RESULT, AND PLOTTING ALL
14 C# THREE SETS OF VALUES.
15 C#
16 C#
17 C# THE RESULTS ARE FOUND FROM THE FOLLOWING APPROXIMATIONS:
18 C#
19 C# EXACT: X=XOSC+CNA(LAMBDA,OMEGA*T)
20 C#
21 C# CHORDS: X=XOSC+(XOC-XOSC)*COS(PT=(T-TOC))
22 C# +(XDOTOC/PC)*SIN(PT=(T-TOC))
23 C#
24 C# X(DOT)=(XOC-XOSC)*SIN(PT=(T-TOC))
25 C# +(XDOTOC*COS(PT=(T-TOC)))
26 C#
27 C# TANGENTS: X=XOST+(XOC1-XOST)*COS(PT=(T-TOT))
28 C# +(XDOTOT/PT)*SIN(PT=(T-TOT))
29 C#
30 C# X(DOT)=PT*(XOC1-XOST)*SIN(PT=(T-TOT))
31 C# +(XDOTOT*COS(PT=(T-TOT)))
32 C#
33 C# THE VARIOUS PARAMETERS ARE EXPLAINED AND CALCULATED IN THE
34 C# PROGRAM.
35 C#####
36 C#
37 C#
38 C#
39 C#
40 C#
41 C#
42 C#
43 C#
44 C#
45 C#
46 C#
47 IMPLICIT REAL*8 (A-H,D-Z)
48 REAL*8 DELTAX(200),TRUC(200),TAUT(200)
49 REAL*8 J,K,LAMBDA,H
50 REAL*8 AMPL0(200),AMPLC(200),AMPLT(200),TE(200),TC(200),TT(200)
51 REAL*8 YC(200),TC(200),TCP(200),YT(200),TTP(200)
52 REAL*8 XA(2),YA(2)
53 REAL*8 HA,HB,HC,VA,VB,YC
54 INTEGER*4 ALPH(20)
55 COMMON/CDEFF/A,B
56 COMMON/ELLIP/PI,LAMBDA,K
57 COMMON/ENDOC/XOC
58 COMMON/ENDIC/XIC
59 COMMON/CROSSC/XOSC
60 COMMON/VIBC/PC
61 COMMON/ENDOT/XOT
62 COMMON/ENDIT/XIT
63 COMMON/CROSST/XOST
64 COMMON/SHARE/XOC2
65 COMMON/VIST/PT
66 C#####
67 C#
68 C# THE FOLLOWING VALUES ARE READ FROM DATA FILE HSPRINGDATA:
69 C#
70 C#
71 C# A,B=AS IN THE EQUATION
72 C#
73 C# DISPL0=INITIAL AMPLITUDE
74 C# VO=INITIAL VELOCITY
75 C# TO=INITIAL TIME
76 C#
77 C# HA,HC=PLOT PARAMETERS FOR HORIZONTAL AXIS (HC AN INTEGRAL MULTIPLE
78 C# OF 8.0)
79 C#
80 C# VC=PLOT PARAMETER FOR VERTICAL AXIS (AN INTEGRAL MULTIPLE OF 7.0,
81 C# 7.0 GIVING BEST RESULTS)
82 C#
83 C# XA,YA=PARAMETERS FOR PLOTTING ZERO LINE
84 C#
85 C# ALPH=ARRAY FOR PLOT AXIS LABELS
86 C#
87 READ(5,10) A,B
88 10 FORMAT(2D10.6)
89 READ(5,20) DISPL0,VO,TO
90 20 FORMAT(3D10.6)
91 READ(5,30) HA,HC
92 30 FORMAT(2F10.4)
93 READ(5,40) VC
94 40 FORMAT(F10.4)
95 READ(5,50) XA(1)
96 50 FORMAT(F10.4)
97 READ(5,60) YA(1),YA(2)
98 60 FORMAT(2F10.4)
99 READ(5,70) (ALPH(I),I=1,12)
100 70 FORMAT(12A4)
101 READ(5,80) (ALPH(I),I=13,18)
102 80 FORMAT(4A4)
103 READ(5,90) (ALPH(I),I=17,20)
104 90 FORMAT(4A4)
105 C#
106 C#
107 C# PI=3.141592653589793
108 108=DISPL0*#2/A
109 C#####

```

```

111 C XOC=AMPLITUDE OF EXACT SOLUTION
112 C LAMBDA=MODULUS (FOR EXACT SOLUTION)
113 C
114 C
115 C#####
116 C K0=DISPLO
117 C LAMBDA=DSORT(0.8/((1.0/BETA)+1.0))
118 C#####
119 C ELLINT=SUBROUTINE FOR CALCULATING THE COMPLETE ELLIPTIC INTEGRAL
120 C OF THE FIRST KIND
121 C
122 C#####
123 C CALL ELLINT
124 C#####
125 C
126 C OMEGAE=EXACT SOLUTION ANGULAR FREQUENCY
127 C TIME=CUMULATIVE TIME FOR EXACT SOLUTION
128 C DELTAT=TIME INCREMENT
129 C
130 C#####
131 C OMEGAE=DSORT(A=(1.0+BETA))
132 C TIME=TO
133 C DELTAT=(100.0+DSORT(A=(1.0+BETA)))
134 DO 100 N=1,101
135 U=OMEGAE*TIME
136 C#####
137 C DJCSYF=SUBROUTINE FOR EVALUATING JACOBIAN ELLIPTIC FUNCTIONS
138 C (SEE UBC MANUAL "UBC FUNCTION" FOR DETAILS)
139 C
140 C#####
141 C CALL DJCSYFIU,LAMBDA,DSN,DCH,DDN,IE,0
142 C#####
143 C AMPLE=ARRAY FOR INSTANTANEOUS AMPLITUDE (FOR PLOTTING)
144 C TE=ARRAY FOR TIME (FOR PLOTTING)
145 C
146 C#####
147 C AMPLE(N)=XOC*BCH
148 C TE(N)=TIME
149 C TIME=TIME+DELTAT
150 C 100 CONTINUE
151 C#####
152 C TAUE=EXACT PERIOD
153 C BIG=LARGEST VALUE OF QUARTER-PERIOD (FOR PLOT SCALING)
154 C
155 C#####
156 C TAUE=4.05K/DSORT(A=(1.0+BETA))
157 C BIG=TAUE/4.0
158 C#####
159 C
160 C THE FOLLOWING DO-LOOP (ENDING AT STEP 280) CALCULATES THE
161 C PIECEWISE LINEAR PERIODS (FOR BOTH METHODS) BASED ON THE
162 C PARTICULAR NUMBER OF SEGMENTS 'CHSE'.
163 C
164 C
165 C NC=NUMBER OF CHORDS (INITIALLY SET AT 5)
166 C NT=NUMBER OF TANGENTS (INITIALLY SET AT 6)
167 C
168 C XOC,XOT=INITIAL DISPLACEMENT FOR EACH SUCCESSIVE SEGMENT, FOR
169 C CHORDS AND TANGENTS, RESPECTIVELY
170 C
171 C XDOTOC,XDOTOT=INITIAL VELOCITY FOR EACH SUCCESSIVE SEGMENT, AS
172 C ABOVE
173 C TOC,TOT=INITIAL TIME, FOR CHORDS AND TANGENTS, RESPECTIVELY
174 C
175 C XOC1=INITIAL CROSSING POINT FOR TANGENT LINES OF TWO ADJACENT
176 C SEGMENTS (SEE SUBROUTINE XSHARE BELOW)
177 C
178 C DELTAP=LENGTH OF SEGMENT
179 C DELTAX=ARRAY FOR DELTA
180 C
181 C#####
182 C NC=5
183 DO 280 I=1,20
184 NT=NC-1
185 H=NC
186 H=NC
187 XOC=DISPLO
188 XOT=DISPLO
189 XDOTOC=V0
190 XDOTOT=V0
191 TOC=T0
192 TOT=T0
193 XOC1=DISPLO
194 DELTAVDISPLO/H
195 DELTAX(1)=DELTA
196 C#####
197 C
198 C THE FOLLOWING SECTION CALCULATES THE PERIOD FOR THE METHOD OF
199 C CHORDS (ENDING AT STEP 110).
200 C
201 C
202 C
203 C
204 C
205 C
206 C
207 C
208 C XIC=DISPLACEMENT AT END OF SEGMENT
209 C
210 C#####
211 C XIC=XOC-DELTA
212 C DO 110 N=1,NC
213 C#####
214 C
215 C XPC=SUBROUTINE FOR CALCULATING POINT WHERE SEGMENT CHORD LINE
216 C CROSSES X-AXIS
217 C
218 C FREQC=SUBROUTINE FOR CALCULATING ANGULAR FREQUENCY
219 C
220 C#####

```

```

221 CALL XPC
222 CALL PREOC
223
224 C THE FOLLOWING SOLVES THE TRIGONOMETRIC EQUATION:
225 C
226 C CSC=C1C=COS(THETAC)+C2C=SIN(THETAC)
227 C
228 C WHERE:
229 C
230 C THETAC=PC(T1C-TOC)
231 C
232 C TIC BEING THE CUMULATIVE TIME TAKEN TO REACH THE END OF THE
233 C SEGMENT BEING EXAMINED.
234 C
235 C
236 C1C=XOC-XOST
237 C2C=RDOTOC/PC
238 C3C=XIC-XOST
239 THETAC=DATAN(C2C/C1C)+DARCCOS(C3C/DSQRT(C1C**2+C2C**2))
240 T1C=TOC+THETAC/PC
241
242 C
243 C AMPLC=ARRAY FOR XOC (FOR PLOTTING)
244 C TCHARRAY FOR TOC (FOR PLOTTING)
245 C
246 C
247 CAMPLC(N)=XOC
248 TC(N)=TOC
249
250 C
251 C RESETTING THE INITIAL CONDITIONS FOR THE NEXT SEGMENT.
252 C
253 C
254 CXC=XIC
255 XIC=XOC-DELTA
256 RDOTOC=-PC*(C1C=COS(THETAC)-C2C=SIN(THETAC))
257 TOC=TIC
258 110 CONTINUE
259
260 C
261 C TAUCHARRAY FOR PERIOD FOUND USING CHORDS
262 C
263 C
264 TAUC(1)=4.0*TIC
265 IF(TIC.GE.BIG) BIG=TIC
266 NDC=NDC+1
267 AMPLC(NDC)=XOC
268 TC(NDC)=TOC
269
270 C
271 C THE FOLLOWING SECTION CALCULATES THE PERIOD FOR THE METHOD OF
272 C TANGENTS (ENDING AT STEP 120).
273 C
274 C
275 C
276 C XIT=DISPLACEMENT AT END OF SEGMENT
277 C
278 C
279 C
280 C
281 XIT=XOT-DELTA
282 DO 120 N=1,NT
283 C
284 C XSHARE=SUBROUTINE FOR CALCULATING THE CROSSING POINT FOR TANGENT
285 C LINES OF TWO ADJACENT SEGMENTS
286 C
287 C XPT=SUBROUTINE FOR CALCULATING POINT WHERE SEGMENT TANGENT LINE
288 C CROSSES X-AXIS
289 C
290 C FREOT=SUBROUTINE FOR CALCULATING ANGULAR FREQUENCY
291 C
292 C
293 C
294 CALL XSHARE
295 CALL XPT
296 CALL FREOT
297
298 C THE FOLLOWING SOLVES THE TRIGONOMETRIC EQUATION:
299 C
300 C3T=C1T=COS(THETAC)+C2T=SIN(THETAC)
301 C
302 C WHERE:
303 C
304 C THETAT=PT(T1T-TOT)
305 C
306 C T1T BEING THE CUMULATIVE TIME TAKEN TO REACH THE END OF THE
307 C SEGMENT BEING EXAMINED.
308 C
309 C
310 C1T=XOC1-XOST
311 C2T=RDOTOT/PT
312 C3T=XOC2-XOST
313 IF(DABS(XOT).LT.1.0D-10) C3T=-XOST
314 THETAT=DATAN(C2T/C1T)+DARCCOS(C3T/DSQRT(C1T**2+C2T**2))
315 T1T=TOT+THETAT/PT
316
317 C
318 C AMPLT=ARRAY FOR XOC1 (FOR PLOTTING)
319 TT=ARRAY FOR TOT (FOR PLOTTING)
320 C
321 C
322 CAMPLT(N)=XOC1
323 TT(N)=TOT
324 C
325 C
326 C RESETTING THE INITIAL CONDITIONS FOR THE NEXT SEGMENT.
327 C
328 C
329 COT=XIT
330

```

```

331     X1T=X0T-DELTA
332     XDOTOT=-PT*(C1T+DSIN(THETAT))-C2T+DCOS(THETAT)
333     TOT=T1T
334     XC1=XOC1
335   120 CONTINUE
336 C#####
337 C      TAUT-PERIOD FOUND USING TANGENTS
338 C#####
339 C      THE NEXT FEW LINES ARE USED IN FINALIZING ALL THE PLOT PARAMETERS
340 C      WHICH INCLUDES AXIS SCALING.
341 C#####
342 C      TAUY(1)=4.0*T1T
343 C      IF(T1T.GE.BIG) BIG=T1T
344 C      NDT=NDT+1
345 C      AMPLT(NDT)=XOC1
346 C      TT(NDT)=TOT
347 C#####
348 C      THE NEXT FEW LINES ARE USED IN FINALIZING ALL THE PLOT PARAMETERS
349 C      WHICH INCLUDES AXIS SCALING.
350 C#####
351 C#####
352 C      M=1
353 C      IF(DISPL0.LE.1.0) GO TO 140
354 130 CONTINUE
355 DECD=DISPL0/(10.0**M)
356 IF(DECD.LE.1.0) GO TO 180
357 M=M+1
358 GO TO 130
359 140 CONTINUE
360 DIGD=DISPL0/(10.0**M)
361 IF(DIGD.LE.1.0) GO TO 180
362 M=M+1
363 GO TO 140
364 150 CONTINUE
365 DECD=DIGD/10.0
366 M=(M-1)
367 160 CONTINUE
368 IF(DECD.LE.0.1) VFACT=0.1
369 IF((DECD.GT.0.1).AND.(DECD.LE.0.25)) VFACT=0.25
370 IF((DECD.GT.0.25).AND.(DECD.LE.0.5)) VFACT=0.5
371 IF((DECD.GT.0.5).AND.(DECD.LE.1.0)) VFACT=1.0
372 VS=(VFACT/(VC-2.0))*10.0**M
373 VAS=VS
374 C#####
375 C      M=1
376 C      IF(BIG.LE.1.0) GO TO 180
377 170 CONTINUE
378 DECT=BIG/(10.0**M)
379 IF(DECT.LE.1.0) GO TO 200
380 M=M+1
381 GO TO 170
382 180 CONTINUE
383 DIGT=BIG/(10.0**M)
384 IF(DIGT.LE.1.0) GO TO 180
385 M=M+1
386 GO TO 180
387 180 CONTINUE
388 DECT=DIGT/10.0
389 M=(M-1)
390 200 CONTINUE
391 IF(DECT.LE.0.08) HFACT=0.1
392 IF((DECT.GT.0.08).AND.(DECT.LE.0.16)) HFACT=0.2
393 IF((DECT.GT.0.16).AND.(DECT.LE.0.4)) HFACT=0.5
394 IF((DECT.GT.0.4).AND.(DECT.LE.0.8)) HFACT=1.0
395 IF((DECT.GT.0.8).AND.(DECT.LE.1.0)) HFACT=2.0
396 HS=HFACT*10.0**((M-1))
397 LONGHC
398 XA(2)=DFLOAT((LONG1+HS-0.005)
399 C#####
400 C      THE FOLLOWING SECTION SETS UP THE CHORD AND TANGENT RESULTS SO
401 C      THAT 8 POINTS OF THE CHORD SOLUTION AND 7 OF THE TANGENT VALUES
402 C      ARE PLOTTED FOR A CLEARER PRESENTATION, WITH THE PLOTS BEING
403 C      ALTERNATED FOR COMPARISON PURPOSES.
404 C#####
405 C      YC=ARRAY FOR PLOTTED CHORD DISPLACEMENTS
406 C      TCP=ARRAY FOR PLOTTED CHORD TIME VALUES
407 C      YT=ARRAY FOR PLOTTED TANGENT DISPLACEMENTS
408 C      TTP=ARRAY FOR PLOTTED TANGENT TIME VALUES
409 C#####
410 C#####
411 INCC=NDC/8
412 INCR=NDT/6
413 INCRT=INCT/2
414 IF(NDT.LT.10) INCRT=1
415 NCC=1
416 NCT=1
417 NPC=1
418 NPT=1
419 210 CONTINUE
420 YC(NPC)=AMPLC(NCC)
421 TCP(NPC)=TC(NCC)
422 IF(NCC.EQ.NDC) GO TO 220
423 NPC=NPC+1
424 NCC=NCC+INCC
425 GO TO 210
426 220 CONTINUE
427 YT(NPT)=AMPLT(NCT)
428 TTP(NPT)=TT(NCT)
429 IF(NCT.EQ.1) GO TO 230
430 NCT=NCT+INCT
431 GO TO 240
432 230 CONTINUE
433 NCT=NCT+INCR
434 240 CONTINUE
435 IF(NCT.GE.NDT) GO TO 250
436 NPT=NPT+1
437 GO TO 220
438 250 CONTINUE
439 NPT=NPT+1

```

```

441 V(T(NPT))=AMPLT(NDT)
442 TTP(NPT)=TT(NDT)
443
444 C PLOTTING THE SOLUTIONS. CONSULT THE WRITEUP ON CGPL/CGPL2 AND
445 C THE MANUAL ON DIGITAL PLOTTING FOR DETAILS.
446
447 C#####
448 ND=101
449 NF=1
450 CALL CGPL2(TE,AMPLE,ND,NF,5,HA,HB,HC,VA,VB,VC,ALPH)
451 ND=NPCT
452 CALL CGPL2(TCP,YC,ND,2,1,HA,HB,NC,VA,VB,VC,ALPH)
453 ND=NPT
454 CALL CGPL2(TTP,YT,ND,3,1,HA,HB,NC,VA,VB,VC,ALPH)
455 NF=4
456 CALL CGPL2(XA,YA,2,NF,4,HA,HB,NC,VA,VB,VC,ALPH)
457 HORIZ=HA
458 VERT=VC+0.5
459 CALL PLOT(HORIZ,VERT,3)
460 HORIZ=HC
461 CALL PLOT(HORIZ,VERT,2)
462 VERT=VERT+6.0
463 CALL PLOT(HORIZ,VERT,2)
464 HORIZ=HA
465 CALL PLOT(HORIZ,VERT,2)
466 VERT=VC+0.5
467 CALL PLOT(HORIZ,VERT,2)
468 STARTX=(RC-7.0)/2.0
469 STARTY=VC+0.5
470 HORIZ=STARTX+0.2
471 VERT=STARTY+4.5
472 CALL SYMBOL(HORIZ,VERT,0.2,'A'+' ',0.0,3)
473 HORIZ=STARTX+1.0
474 CALL NUMBER(HORIZ,VERT,0.2,A,0.0,-6)
475 HORIZ=STARTX+3.5
476 CALL SYMBOL(HORIZ,VERT,0.2,'S'+' ',0.0,3)
477 HORIZ=STARTX+4.2
478 CALL NUMBER(HORIZ,VERT,0.2,S,0.0,-6)
479 HORIZ=STARTX+0.2
480 VERT=STARTY+3.0
481 CALL SYMBOL(HORIZ,VERT,0.2,'NO. OF SEGMENTS (APPROX. TO EXACT)'',0.0
482 ,33)
483 HORIZ=STARTX+0.6
484 VERT=STARTY+3.6
485 CALL SYMBOL(HORIZ,VERT,0.2,'SOLN.'+' ',0.0,12)
486 HORIZ=STARTX+0.2
487 VERT=STARTY+3.2
488 CALL SYMBOL(HORIZ,VERT,0.2,'NO. OF CHORDS ''',0.0,18)
489 HORIZ=STARTX+3.4
490 CPOINT=DFLOAT(NC)
491 CALL NUMBER(HORIZ,VERT,0.2,CPOINT,0.0,-1)
492 HORIZ=STARTX+0.2
493 VERT=STARTY+2.8
494 CALL SYMBOL(HORIZ,VERT,0.2,'NO. OF TANGENTS ''',0.0,17)
495 HORIZ=STARTX+3.8
496 TPOINT=DFLOAT(NT)
497 CALL NUMBER(HORIZ,VERT,0.2,TPOINT,0.0,-1)
498 HORIZ=STARTX+0.2
499 VERT=STARTY+2.0
500 CALL SYMBOL(HORIZ,VERT,0.2,'X(O)'+' ',0.0,6)
501 HORIZ=STARTX+1.8
502 CALL NUMBER(HORIZ,VERT,0.2,DISPL0,0.0,4)
503 HORIZ=STARTX+3.8
504 CALL SYMBOL(HORIZ,VERT,0.2,'X(O)'+' ',0.0,6)
505 HORIZ=STARTX+3.925
506 VERT=STARTY+2.3
507 CALL SYMBOL(HORIZ,VERT,0.2,75,0.0,-1)
508 HORIZ=STARTX+4.8
509 VERT=STARTY+2.0
510 CALL NUMBER(HORIZ,VERT,0.2,V0,0.0,4)
511 HORIZ=STARTX+0.3
512 VERT=STARTY+1.1
513 CALL PLOT(HORIZ,VERT,3)
514 HORIZ=STARTX+0.8
515 CALL PLOT(HORIZ,VERT,2)
516 HORIZ=STARTX+1.0
517 VERT=STARTY+1.0
518 CALL SYMBOL(HORIZ,VERT,0.2,'APPROX. TO EXACT-SOLN.'',0.0,22)
519 HORIZ=STARTX+0.8
520 VERT=STARTY+0.7
521 CALL SYMBOL(HORIZ,VERT,0.2,1,0.0,-1)
522 HORIZ=STARTX+1.0
523 VERT=STARTY+0.8
524 CALL SYMBOL(HORIZ,VERT,0.2,'CHORDS'',0.0,6)
525 HORIZ=STARTX+0.8
526 VERT=STARTY+0.3
527 CALL SYMBOL(HORIZ,VERT,0.2,2,0.0,-1)
528 HORIZ=STARTX+1.0
529 VERT=STARTY+0.2
530 CALL SYMBOL(HORIZ,VERT,0.2,'TANGENTS'',0.0,8)
531 NF=0
532 CALL CGPL2(XA,YA,2,NF,4,HA,HB,NC,VA,VB,VC,ALPH)
533
534 C INCREMENTING THE NUMBER OF LINEAR SEGMENTS.
535 C
536 NC=NC+6
537 280 CONTINUE
538
539 C THE CALCULATED VALUES ARE WRITTEN INTO OUTPUT FILE HSPRINGNUM.
540 C
541 C#####
542 WRITE(10,270)
543 270 FORMAT('1.' 'PERIODS FOR THE EQUATION:')
544 WRITE(10,280)
545 280 FORMAT('0.' 'X(DOUBLE-DOT)+A*X+B*(I-3)*2')
546 WRITE(10,290)
547
```

POOR COPY
COPIE DE QUALITEE INFERIEURE

```

561 280 FORMAT('0','OBTAINED BY PIECEWISE UNILINEARIZATION')
562 WRITE(10,300) A,B
563 300 FORMAT('A=','B=','E20.10,2X,'B=' ,E20.10)
564 WRITE(10,310) DISPL0,Y0
565 310 FORMAT('0','(10) ',E20.10,2X,'(X(DOT)(0)=',E20.10)
566 / WRITE(10,320) TAUS
567 320 FORMAT('0','EXACT SOLUTION PERIOD= ',E20.10)
568 WRITE(10,330)
569 330 FORMAT('0','DELTA X',16X,'CHORDS',16X,'TANGENTS')
570 DD 360 16X,20
571 WRITE(10,340) DELTAX(I),TAUC(I),TAUT(I)
572 340 FORMAT(1X,3(E20.10,2X))
573 CONTINUE
574 STOP
575 END
576 C
577 C
578 C THE FOLLOWING SUBROUTINE CALCULATES THE FIRST ONE HUNDRED TERMS OF
579 C AN INFINITE SERIES APPROXIMATION OF THE COMPLETE ELLIPTIC INTEGRAL
580 C OF THE FIRST KIND. SEE SECT. 17.3, P. 591, "HANDBOOK OF
581 C MATHEMATICAL FUNCTIONS WITH FORMULAS, GRAPHS AND MATHEMATICAL
582 C TABLES", M. ABRAMOWITZ AND I. A. STEGUN, FOR DETAILS.
583 C
584 K=CALCULATED VALUE
585 C
586 C
587 C SUBROUTINE ELLINT
588 C IMPLICIT REAL*8 (A-H,O-Z)
589 C REAL*8 J,K,LAMBDA,M
590 C COMMON/ELLIP/PI,LAMBDA,K
591 C M,LAMBDA,M**2
592 C COEFF=1.0/2.0
593 SUM=0.25*M
594 DO 300 N=1,98
595 J=N+1.0
596 COEFF=COEFF*(2.0*(J-1.0)/(2.0*(J
597 TERM=(COEFF*(J-2)*(M**2*(N+1)))
598 SUM=SUM+TERM
599 300 CONTINUE
600 K=(PI/2.0)*(1.0+SUM)
601 RETURN
602 END
603 C
604 C
605 C
606 C
607 C
608 C THE FOLLOWING SUBROUTINE CALCULATES THE POINT WHERE THE INTERVAL
609 C CHORD LINE WOULD CROSS THE X-AXIS.
610 C
611 C PC=CALCULATED VALUE
612 C XOSC=CALCULATED VALUE
613 C
614 C
615 C SUBROUTINE XPC
616 C IMPLICIT REAL*8 (A-H,O-Z)
617 C COMMON/COEFF/A,B
618 C COMMON/ENDOC/XOC
619 C COMMON/ENDIC/XIC
620 C COMMON/CROSSC/XOSC
621 C FO=XOC+B*(XOC**3)
622 C F1=XIC+B*(XIC**3)
623 C XOSC=(XIC-(F1/FO)*XOC)/(1-(F1/FO))
624 C RETURN
625 C
626 C
627 C
628 C
629 C
630 C
631 C
632 C THE FOLLOWING SUBROUTINE CALCULATES THE ANGULAR FREQUENCY FOR
633 C AN INTERVAL USING CHORDS.
634 C
635 C
636 C SUBROUTINE FREOC
637 C IMPLICIT REAL*8 (A-H,O-Z)
638 C COMMON/COEFF/A,B
639 C COMMON/ENDOC/XOC
640 C COMMON/ENDIC/XIC
641 C COMMON/VIB/PC
642 C FO=XOC+B*(XOC**3)
643 C F1=XIC+B*(XIC**3)
644 C SLOPE=(FO-F1)/(XOC-XIC)
645 C PCDSORT(SLOPE)
646 C RETURN
647 C
648 C
649 C
650 C
651 C
652 C THE FOLLOWING SUBROUTINE CALCULATES THE POINT WHERE THE TANGENT
653 C LINES OF TWO ADJACENT INTERVALS WOULD CROSS.
654 C
655 C XOC2=CALCULATED VALUE
656 C
657 C
658 C
659 C SUBROUTINE XSHARE
660 C IMPLICIT REAL*8 (A-H,O-Z)
661 C COMMON/COEFF/A,B

```

```

661 COMMON/ENDOT/XOT
662 COMMON/ENDIT/XIT
663 COMMON/SHARE/XOC1
664 IF(DBASE(XOT).LT.1.0D-10) GO TO 400
665 SLOPExA=2.0D0*(XOT**2)
666 SLOPEyA=2.0D0*(XIT**2)
667 FORA=XOT+0.001(XOT**2)
668 PI=xATIT+0.001(XIT**2)
669 XOC2=(SLOPExA*XOT-SLOPEyA*XIT-(PI-F1))/(SLOPExA-SLOPEyA)
670 GO TO 410
671 400 CONTINUE
672 XOC2=0.00
673 410 CONTINUE
674 RETURN
675 END
676 C
677 C
678 C
679 C*****CALCULATES THE POINT WHERE THE TANGENT LINE WOULD CROSS THE X-AXIS.
680 C
681 C THE FOLLOWING SUBROUTINE CALCULATES THE POINT WHERE THE INTERVAL
682 C TANGENT LINE WOULD CROSS THE X-AXIS.
683 C
684 C XOST=CALCULATED VALUE
685 C
686 C*****SUBROUTINE XPT
687 IMPLICIT REAL*8 (A-H,D-Z)
688 COMMON/CBEFF/A,B
689 COMMON/ENDOT/XOT
690 COMMON/ENDIT/XIT
691 COMMON/CROSSST/XOST
692 F=A*XOT+B*(XOT**2)
693 SLOPExA=3.0D0*(XOT**2)
694 XOST=XOT-F/SLOPE
695 RETURN
696 END
697 C
698 C
699 C
700 C*****CALCULATES THE ANGULAR FREQUENCY FOR
701 C
702 C THE FOLLOWING SUBROUTINE CALCULATES THE ANGULAR FREQUENCY FOR
703 C AN INTERVAL USING TANGENTS.
704 C
705 C
706 C PT=CALCULATED VALUE
707 C
708 C*****SUBROUTINE FREOT
709 IMPLICIT REAL*8 (A-H,D-Z)
710 COMMON/COEFF/A,B
711 COMMON/ENDOT/XOT
712 COMMON/ENDIT/XIT
713 COMMON/VIBT/PT
714 SLOPExA=3.0D0*(XOT**2)
715 PT=DSQRT(SLOPE)
716 RETURN
717 END
718
719 END OF FILE

```

POOR COPY
COPIE DE QUALITEE INFERIEURE

```

1.      ****
2.      C*
3.      C*      PROGRAM HSPRINGC
4.      C*
5.      C*
6.      C*      THE FOLLOWING PROGRAM CALCULATES THE SOLUTION FOR THE *
7.      C*      EQUATION:
8.      C*
9.      C*      X(DOUBLE-DOT)+C*X(DOT)+A*X+B*(X^2)=0
10.     C*
11.     C*      OBTAINED BY PIECEWISE LINEARIZATION USING CHORDS, AS WELL AS
12.     C*      FINDING THE FIRST TROUGH ON THE DISPLACEMENT-TIME CURVE.
13.     C*
14.     C*
15.     C*      THE RESULTS ARE FOUND FROM THE FOLLOWING:
16.     C*
17.     C*      X=XSTAR+EXP(-N*(T-TO))=(C1=COS(PSTAR*(T-TO))
18.     C*                  +C2=SIN(PSTAR*(T-TO)))
19.     C*
20.     C*      X(DOT)=EXP(-N*(T-TO))=(C3=COS(PSTAR*(T-TO))
21.     C*                  +C4=SIN(PSTAR*(T-TO)))
22.     C*
23.     C*      C1=X0-XSTAR
24.     C*      C2=(N*(X0-XSTAR)+XDBTO)/P
25.     C*      C3=N*C1+P*C2
26.     C*      C4=P*C1-N*C2
27.     C*
28.     C*      THE VARIOUS PARAMETERS ARE EXPLAINED AND CALCULATED IN THE
29.     C*      PROGRAM.
30.     C*
31.     C*
32.     C*      THE PROGRAM ALSO SOLVES THE EQUATION USING FIFTH- AND SIXTH-
33.     C*      ORDER RUNGE-KUTTA METHODS. BOTH SOLUTIONS ARE PLOTTED.
34.     C*
35.     C*
36.     C*
37.     C*
38.     C*
39.     C*
40.     C*
41.     C*
42.     C*
43.     C*
44.     C*
45.     C*
46.     C*
47.     C*      IMPLICIT REAL*8 (A-H,O-Z)
48.     C*      EXTERNAL FCN
49.     C*      REAL*8 Y(2),CN(24),W(2,20)
50.     C*      REAL*8 N
51.     C*      REAL*4 AMPLP(200),AMPLRK(200),TP(200),TRK(200)
52.     C*      REAL*4 YP(2000),TPP(200)
53.     C*      REAL*4 XA(2),YA(2)
54.     C*      REAL*4 HA,HG,HC,YA,YB,YC
55.     C*      INTEGER*4 ALPH(20)
56.     C*      COMMON/CDEEFF/A,B
57.     C*      COMMON/DAMP/C
58.     C*****
59.     C*      THE FOLLOWING VALUES ARE READ FROM DATA FILE HSPRINGC DATA:
60.     C*
61.     C*
62.     C*      A,B,C SAME AS ABOVE
63.     C*
64.     C*      DISPL0=INITIAL AMPLITUDE
65.     C*      V0=INITIAL VELOCITY
66.     C*      TIME0=INITIAL TIME
67.     C*
68.     C*      N=NUMBER OF DIVISIONS OF INITIAL DISPLACEMENT
69.     C*      DTIME=TIME INCREMENT (RUNGE-KUTTA SOLUTION)
70.     C*
71.     C*      HA,HG=PLOT PARAMETERS FOR HORIZONTAL AXIS (HG AN INTEGRAL MULTIPLE
72.     C*          OF 7.0)
73.     C*
74.     C*      VC=PLOT PARAMETER FOR VERTICAL AXIS (AN INTEGRAL MULTIPLE OF 12.0,
75.     C*          12.0 GIVING BEST RESULTS)
76.     C*
77.     C*      XA,YA=PARAMETERS FOR PLOTTING ZERO LINE
78.     C*
79.     C*      ALPH=ARRAY FOR PLOT AXIS LABELS
80.     C*
81.     C*****
82.     C*      READ(5,10) A,B,Ω
83.     10 FORMAT(3D10.6)
84.     C*      READ(5,20) DISPL0,V0,TIME0
85.     20 FORMAT(3D10.6)
86.     C*      READ(5,30) N,DTIME
87.     30 FORMAT(2D10.6)
88.     C*      READ(5,40) HA,HC
89.     40 FORMAT(2F10.6)
90.     C*      READ(5,50) VC
91.     50 FORMAT(F10.4)
92.     C*      READ(5,60) XA(1)
93.     60 FORMAT(F10.4)
94.     C*      READ(5,70) YA(1),YA(2)
95.     70 FORMAT(2F10.4)
96.     C*      READ(5,80) (ALPH(I),I=1,12)
97.     80 FORMAT(12A4)
98.     C*      READ(5,90) (ALPH(I),I=13,18)
99.     90 FORMAT(4A4)
100.    C*      READ(5,100) (ALPH(I),I=17,20)
101.    100 FORMAT(4A4)
102.    C*****
103.    C*      N=DAMPING FACTOR
104.    C*
105.    C*      HG=HC/2.0
106.    C*****
107.    C*****
```

```

111 C      SETTING THE INITIAL CONDITIONS.
112 C
113 C      X0=INITIAL DISPLACEMENT FOR EACH SUCCESSIVE SEGMENT
114 C      XDOT0=INITIAL VELOCITY FOR EACH SEGMENT
115 C      TO=CUMULATIVE TIME TAKEN TO REACH BEGINNING OF SEGMENT
116 C      DELT=INITIAL GUESS AT TIME INTERVAL FOR THE SEGMENT-BEING EXAMINED
117 C      (SEE SUBROUTINE TIME BELOW)
118 C
119 C      DELTAX=LENGTH OF SEGMENT
120 C
121 C      I= COUNTER FOR DATA POINTS TO BE PLOTTED
122 C
123 C#####
124 C      X0=DISPL0
125 C      XDOT0=V0
126 C      TO=TIME0
127 C      DELT=T0
128 C      DELTAX=DISPL0/H
129 C      DELT=DELTAX
130 C      I=0
131 C#####
132 C
133 C      THE EQUATION PARAMETERS ARE WRITTEN INTO OUTPUT FILE HSPRINGCHUM.
134 C
135 C#####
136 C      WRITE(10,130)
137 C      110 FORMAT('1', 'PIECEWISE LINEAR PERIOD FOR THE EQUATION:', )
138 C      WRITE(10,120)
139 C      120 FORMAT('0', 'X(DOUBLE-DOT)+C=X(DOT)+A*X+B=(I=2)=0')
140 C      WRITE(10,130) A,B,C
141 C      130 FORMAT('0', 'A', '0', 'G20.10', '2X', 'B', 'G20.10', '2X', 'C', 'G20.10')
142 C      WRITE(10,140) DISPL0,V0
143 C      140 FORMAT('0', 'X(0)', 'G20.10', '2X', '(X(DOT))(0)', 'G20.10')
144 C      WRITE(10,150)
145 C      150 FORMAT(' ', 'X', 'AMPLITUDE', 'Y0K', 'CUMULATIVE TIME')
146 C      150 CONTINUE
147 C      I=I+1
148 C#####
149 C
150 C      X1=DISPLACEMENT AT END OF SEGMENT
151 C
152 C      XPRIME=SUBROUTINE FOR CALCULATING POINT WHERE SEGMENT CHORD LINE
153 C      CROSSES X-AXIS
154 C      XSTAR=CALCULATED VALUE
155 C
156 C      FREQ=SUBROUTINE FOR CALCULATING ANGULAR FREQUENCY
157 C      PSTAR=CALCULATED VALUE
158 C
159 C      TIME=SUBROUTINE FOR CALCULATING TIME INTERVAL FOR A SEGMENT
160 C      DELTAT=CALCULATED VALUE
161 C
162 C      SPEED=SUBROUTINE FOR CALCULATING SPEED AT END OF SEGMENT
163 C      XDOT1=CALCULATED VALUE
164 C
165 C--NOTE: THE PROGRAM STOPS IF THE DISPLACEMENT AT THE END OF A SEGMENT
166 C      IS EQUAL TO -DISPL (I. E., UNDAMPED CASE).
167 C
168 C#####
169 C      X1=X0-DELTAX
170 C      CALL XPRIME(X0,X1,XSTAR)
171 C      CALL FREQ(X0,X1,N,PSTAR)
172 C      CALL TIME(X0,XDOT0,X1,DELT,XSTAR,N,PSTAR,DELTAT)
173 C      CALL SPEED(X0,XDOT0,XSTAR,N,DELTAT,PSTAR,XDOT1)
174 C#####
175 C
176 C      STARTING AT STEP 180 AND CONTINUING AT 200, THE PROGRAM TESTS FOR
177 C      ZERO VELOCITY. IF IT OCCURS AT THE END OF THE SEGMENT, THE
178 C      PROGRAM STOPS (SEE STEP 300). IF NOT, THE LOCATION OF THE FIRST
179 C      TROUGH IS SOUGHT (STEPS 250 TO 270).
180 C
181 C
182 C      DELT=DELTAT
183 C      T1=TO+DELTAT
184 C      IF(X0.GE.0.0) GO TO 170
185 C      GO TO 180
186 C      170 CONTINUE
187 C#####
188 C
189 C      AMPLP=ARRAY FOR X0
190 C      TP=ARRAY FOR TO
191 C
192 C#####
193 C      AMPLP(I)=X0
194 C      TP(I)=TO
195 C#####
196 C
197 C      THE INTERIM VALUES ARE WRITTEN INTO OUTPUT FILE HSPRINGCHUM.
198 C
199 C#####
200 C      WRITE(10,180) X0,TO
201 C      180 FORMAT(1X,2(G20.10,2X))
202 C
203 C
204 C      RESETTING THE INITIAL CONDITIONS.
205 C
206 C#####
207 C      X0=X1
208 C      XDOT0=XDOT1
209 C      TO=TI
210 C      GO TO 180
211 C      180 CONTINUE
212 C      IF(DABS(XDOT1).LT.1.0D-08) GO TO 200
213 C      GO TO 170
214 C      200 CONTINUE
215 C
216 C
217 C      THE SEGMENT IN WHICH THE FIRST TROUGH OCCURS IS WRITTEN INTO
218 C      OUTPUT FILE HSPRINGCHUM.
219 C
220 C#####

```

```

221      AMPLP(I)=X0
222      TP(I)=TO
223      WRITE(10,210) X0,TO
224      210 FORMAT(1X,2(G20.10,2X))
225      DIFF=DISPL0-X0
226      IF(DABS(DIFF).LT.1.0D-08) GO TO 300
227      WRITE(10,220) X1,T1
228      220 FORMAT(1X,2(G20.10,2X))
229      WRITE(10,230)
230      230 FORMAT(' ',/ 'THROUGH ITERATIONS')
231      WRITE(10,240)
232      240 FORMAT('0',1GX,'X0',20X,'X1',14X,'X1(CALCULATED)')
233      C#####
234      C      THE FOLLOWING (ENDING AT STEP 270) IS THE CALCULATION OF THE FIRST
235      C      TROUGH. USING THE TIME INTERVAL REQUIRED TO REACH ZERO VELOCITY,
236      C      A NEW VALUE OF X1 (XF) IS CALCULATED AND COMPARED TO THE PREVIOUS
237      C      VALUE. THIS NEW VALUE BECOMES THE NEW X1 IF THE DIFFERENCE
238      C      BETWEEN XF AND THE OLD X1 IS NOT WITHIN A SET TOLERANCE, AND THE
239      C      PROCESS STARTS AGAIN.
240      C
241      C      THESE ITERATIONS ARE WRITTEN INTO OUTPUT FILE HSPRINGCHUM.
242      C
243      C#####
244      250 CONTINUE
245      C      X=X0-XSTAR
246      C      C2=(M=(X0-XSTAR)+XDOT0)/PSTAR
247      C      P=PSTAR
248      C      XF=XSTAR+DEXP(-N*DELTAT)=(C1=DCOS(P*DELTAT)+C2=DSIN(P*DELTAT))
249      C      WRITE(10,260) X0,X1,XF
250      C      260 FORMAT(1X,3(G20.10,2X))
251      C      DIFF=X1-XF
252      C      IF(DABS(DIFF).LT.1.0D-08) GO TO 270
253      C      X1=XF
254      C      DIFF=X0-XF
255      C      IF(DABS(DIFF).LT.1.0D-08) GO TO 270
256      C      CALL XPRIME(X0,X1,M,PSTAR)
257      C      CALL FREQ(X0,X1,M,PSTAR)
258      C      CALL TIME(X0,XDOT0,X4,DELT,XSTAR,M,PSTAR,DELTAT)
259      C      GO TO 280
260
261      270 CONTINUE
262      C#####
263      C      THE FINAL VALUES ARE WRITTEN INTO OUTPUT FILE HSPRINGCHUM.
264      C
265      C#####
266      280 T=TP(I)
267      X=X1
268      I=I+1
269      NPOINT=I
270      AMPLP(I)=XF
271      TP(I)=TF
272      TAU=2.0*T
273      TAU=2.0*T
274      WRITE(10,280) DELTAX,TAU
275      280 FORMAT(' ',/ 'DELTAX',G20.10,2X,'PERIOD',G20.10)
276      WRITE(10,290) XF
277      290 FORMAT('0',/ 'FIRST TROUGH OCCURS AT X*',G20.10)
278      GO TO 320
279
280      300 CONTINUE
281      C#####
282      C      THE FINAL VALUES ARE WRITTEN INTO OUTPUT FILE HSPRINGCHUM (UNDAMPED
283      C      CASE).
284      C
285      C#####
286      310 NPOINT=1
287      TAU=2.0*T
288      WRITE(10,310) DELTAX,TAU
289      310 FORMAT(' ',/ 'DELTAX',G20.10,2X,'PERIOD',G20.10)
290      320 CONTINUE
291      C#####
292      C      THE FOLLOWING SECTION (ENDING AT LINE 330) IS THE RUNGE-KUTTA
293      C      SOLUTION OF THE SAME ORIGINAL EQUATION.
294      C
295      C
296      C
297      C      TFINAL=FINAL TIME
298      C      INTVL=NUMBER OF INCREMENTS
299      C
300      C#####
301      310 TFINAL=T1+O_1
302      INTVL=TFINAL/DTIME+1
303      C
304      C
305      C      THE FOLLOWING ARE PARAMETERS FOR THE NUMERICAL SUBROUTINE.
306      C      CONSULT THE IMSL MANUAL ON "OVERK" FOR FURTHER DETAILS.
307      C
308      C#####
309      XRK=TIME0
310      Y(1)=DISPL0
311      Y(2)=V0
312      NEON=2
313      NW=2
314      TOL=1.0D-12
315      IND=1
316      XEND=DTIME
317      C
318      C      ZEIT=CUMULATIVE VALUE OF TIME AT END OF EACH INCREMENT
319      C
320      C
321      C
322      C      ZEIT=0.0
323      C
324      C
325      C      AMPLRK=ARRAY FOR AMPLITUDE VALUES
326      C      TRK=ARRAY FOR TIME VALUES
327      C
328      C#####
329      330 DO 330 J=1,INTVL

```

```

331      AMPLR(J)=Y(1)
332      TRK(J)=ZKIT
333      CALL DVERK(NEON,FCN,XRK,Y,XEND,TOL,IND,CH,NW,W,IER)
334
335      C      INCREMENTING THE TIME VALUES.
336
337      C      ZKIT=XEND
338      XEND=XEND+DTIME
339      330 CONTINUE
340
341      C      THE CALCULATED VALUES ARE WRITTEN INTO OUTPUT FILE MSPRINGCRK.
342
343      C
344      C      THE NEXT FEW LINES ARE USED IN FINALIZING ALL THE PLOT PARAMETERS
345      C      WHICH INCLUDES AXIS SCALING.
346
347      WRITE(11,340)
348      340 FORMAT('1', 'RUNGE-KUTTA RESULTS FOR THE EQUATION:')
349      WRITE(11,350)
350      350 FORMAT('0', 'X(DOUBLE-DOT)+C*X(DOT)+A*X+B=(X**3)=0')
351      WRITE(11,350) A,B,C
352      350 FORMAT('0', 'A=' , 820.10,2X,'B=' , 820.10,2X,'C=' , 820.10)
353      WRITE(11,370) DISPL0,Y0
354      370 FORMAT('0', 'X(0)' , 820.10,2X,'X(DOT)(0)' , 820.10)
355      WRITE(11,380)
356      380 FORMAT('---',3X,'TIME',18X,'AMPLITUDE')
357      DO 400 NOW1,INTVL
358      WRITE(11,380) TRK(NOW1),AMPLR(NOW1)
359      380 FORMAT(1X,2(820.10,2X))
360      400 CONTINUE
361
362      C
363      C      THE NEXT FEW LINES ARE USED IN FINALIZING ALL THE PLOT PARAMETERS
364      C      WHICH INCLUDES AXIS SCALING.
365      C
366
367      M=1
368      IF(DISPL0.LE.1.0) GO TO 420
369      410 CONTINUE
370      DECD=DISPL0/(10.0**M)
371      IF(DECD.LE.1.0) GO TO 440
372      M=M+1
373      GO TO 410
374      420 CONTINUE
375      DIGD=DISPL0*(10.0**M)
376      IF(DIGD.GE.1.0) GO TO 430
377      M=M+1
378      GO TO 420
379      430 CONTINUE
380      DECD=DIGD/10.0
381      M=M-1
382      440 CONTINUE
383      IF(DECD.LE.0.1) VFACT=0.1
384      IF((DECD.GT.0.1).AND.(DECD.LE.0.25)) VFACT=0.25
385      IF((DECD.GT.0.25).AND.(DECD.LE.0.5)) VFACT=0.5
386      IF((DECD.GT.0.5).AND.(DECD.LE.1.0)) VFACT=1.0
387      SFAC=VC/10.0
388      VAF=VFACT*SFAC*(10.0**M)
389      DEN=VC/(2.0*SFAC)
390      VBN=(VFACT/DEN)*10.0**M
391
392      M=1
393      IF(TFINAL.LE.1.0) GO TO 460
394      450 CONTINUE
395      DECT=TFINAL/(10.0**M)
396      IF(DECT.LE.1.0) GO TO 460
397      M=M+1
398      GO TO 450
399      460 CONTINUE
400      DIGT=TFINAL*(10.0**M)
401      IF(DIGT.GE.1.0) GO TO 470
402      M=M+1
403      GO TO 460
404      470 CONTINUE
405      DECT=DIGT/10.0
406      M=M-1
407      480 CONTINUE
408      IF(DECT.LE.0.07) HFACT=0.1
409      IF((DECT.GT.0.07).AND.(DECT.LE.0.14)) HFACT=0.2
410      IF((DECT.GT.0.14).AND.(DECT.LE.0.35)) HFACT=0.5
411      IF((DECT.GT.0.35).AND.(DECT.LE.0.7)) HFACT=1.0
412      IF((DECT.GT.0.7).AND.(DECT.LE.1.0)) HFACT=2.0
413      HBN=HFACT*(10.0**M-1)
414      LONG=HC
415      XA(2)=DFLOAT(LONG)-HB-0.005
416
417      C      THE FOLLOWING SECTION SETS UP THE PIECEWISE-LINEAR RESULTS SO THAT
418      C      NO MORE THAN 13 POINTS ARE PLOTTED FOR A CLEAER-PRESENTATION.
419
420      C
421      C      YP=ARRAY FOR PLOTTED PIECEWISE LINEAR DISPLACEMENTS
422      C      TPP=ARRAY FOR PLOTTED PIECEWISE LINEAR TIME VALUES
423
424
425      COUNT=DFLOAT(NPOINT)
426      INC=COUNT/10.0
427      DELINC=COUNT*(10.0-DFLOAT(INC))
428      IF(DELINC.GE.0.8) INC=INC+1
429      NP=1
430      ND=1
431      480 CONTINUE
432      YP(NP)=AMPLP(ND)
433      TPP(NP)=TP(ND)
434      NP=NP+1
435      ND=ND+INC
436      IF(ND.GE.NPOINT) GO TO 500
437      GO TO 480
438      500 CONTINUE
439      YP(NP)=AMPLP(NPOINT)
440      TPP(NP)=TP(NPOINT).

```

```

441 C#####
442 C PLOTTING THE SOLUTIONS... CONSULT THE WRITEUP ON CGPL/CGPL2 AND
443 C THE MANUAL ON DIGITAL PLOTTING FOR DETAILS.
444 C
445 C#####
446 ND=INTVL
447 NF=1
448 CALL CGPL2(TRK,AMPLR,ND,NF,1,HA,HB,HC,VA,VB,VC,ALPH)
449 ND=NP
450 CALL CGPL2(TPP,YP,ND,2,1,HA,HB,HC,VA,VB,VC,ALPH)
451 NF=4
452 CALL CGPL2(XA,YA,2,NF,4,HA,HB,HC,VA,VB,VC,ALPH)
453 STARTX=HC-.3/2.0
454 STARTY=VC+0.5
455 HORIZ=HA
456 VERT=VC+0.5
457 CALL PLOT(HORIZ,VERT,3)
458 HORIZ=HC
459 CALL PLOT(HORIZ,VERT,2)
460 VERT=VERT+4.8
461 CALL PLOT(HORIZ,VERT,2)
462 HORIZ=HA
463 CALL PLOT(HORIZ,VERT,2)
464 VERT=VC+0.8
465 CALL PLOT(HORIZ,VERT,2)
466 HORIZ=STARTX+0.2
467 VERT=STARTY+4.2
468 CALL SYMBOL(HORIZ,VERT,0.2,'A','=','0.0,3)
469 HORIZ=STARTX+1.0
470 CALL NUMBER(HORIZ,VERT,0.2,A,0.0,0)
471 HORIZ=STARTX+2.0
472 CALL SYMBOL(HORIZ,VERT,0.2,'B','=','0.0,3)
473 HORIZ=STARTX+3.8
474 CALL NUMBER(HORIZ,VERT,0.2,B,0.0,0)
475 HORIZ=STARTX+2.0
476 VERT=STARTY+2.8
477 CALL SYMBOL(HORIZ,VERT,0.2,'C','=','0.0,3)
478 HORIZ=STARTX+2.8
479 CALL NUMBER(HORIZ,VERT,0.2,C,0.0,0)
480 HORIZ=STARTX+0.2
481 VERT=STARTY+3.1
482 CALL SYMBOL(HORIZ,VERT,0.2,B2,0.0,-1)
483 HORIZ=STARTX+0.4
484 CALL SYMBOL(HORIZ,VERT,0.2,'X (PIECEWISE)',0.0,11)
485 HORIZ=STARTX+1.0
486 VERT=STARTY+2.8
487 CALL SYMBOL(HORIZ,VERT,0.2,'LINEARIZATION',0.0,16)
488 HORIZ=STARTX+4.4
489 CALL NUMBER(HORIZ,VERT,0.2,DELX,0.0,4)
490 HORIZ=STARTX+0.2
491 VERT=STARTY+2.4
492 CALL SYMBOL(HORIZ,VERT,0.2,B2,0.0,-1)
493 HORIZ=STARTX+0.4
494 CALL SYMBOL(HORIZ,VERT,0.2,'T (RUNGE-KUTTA)',0.0,17)
495 HORIZ=STARTX+4.0
496 CALL NUMBER(HORIZ,VERT,0.2,DTIME,0.0,4)
497 HORIZ=STARTX+0.2
498 VERT=STARTY+1.8
500 CALL SYMBOL(HORIZ,VERT,0.2,'X(0)','=','0.0,6)
501 HORIZ=STARTX+1.8
502 CALL NUMBER(HORIZ,VERT,0.2,DISPL0,0.0,6)
503 HORIZ=STARTX+2.8
504 CALL SYMBOL(HORIZ,VERT,0.2,'X(0)','=','0.0,6)
505 HORIZ=STARTX+3.828
506 VERT=STARTY+1.9
507 CALL SYMBOL(HORIZ,VERT,0.2,78,0.0,-1)
508 HORIZ=STARTX+4.0
509 VERT=STARTY+1.6
510 CALL NUMBER(HORIZ,VERT,0.2,V0,0.0,0)
511 HORIZ=STARTX+0.3
512 VERT=STARTY+0.7
513 CALL PLOT(HORIZ,VERT,3)
514 HORIZ=STARTX+0.8
515 CALL PLOT(HORIZ,VERT,2)
516 HORIZ=STARTX+1.0
517 VERT=STARTY+0.8
518 CALL SYMBOL(HORIZ,VERT,0.2,'RUNGE-KUTTA',0.0,11)
519 HORIZ=STARTX+0.8
520 VERT=STARTY+0.3
521 CALL SYMBOL(HORIZ,VERT,0.2,1,0.0,-1)
522 HORIZ=STARTX+1.0
523 VERT=STARTY+0.3
524 CALL SYMBOL(HORIZ,VERT,0.2,'PIECEWISE LINEARIZATION',0.0,23)
525 NF=0
526 CALL CGPL2(XA,YA,2,NF,4,HA,HB,HC,VA,VB,VC,ALPH)
527 STOP
528 END
529 C
530 C
531 C
532 C
533 C
534 C
535 C
536 C
537 C
538 C#####
539 C THE FOLLOWING SUBROUTINE CALCULATES THE POINT WHERE THE INTERVAL
540 C CHORD LINE WOULD CROSS THE X-AXIS.
541 C
542 C#####
543 SUBROUTINE XPRIME(X0,X1,XSTAR)
544 IMPLICIT REAL*8 (A-H,O-Z)
545 COMMON/COEFF/A,B
546 IF(DABS(X0).LT.1.0D-08) GO TO 500
547 F0=A*X0+B*(X0**2)
548 F1=A*X1+B*(X1**2)

```

```

561      XSTAR=(X1-(P1/P0)*X0)/(1-(P1/P0))
562      GO TO 610
563 600 CONTINUE
564      XSTAR=0.0
565 610 CONTINUE
566      RETURN
567      END *
568
569 C
570 C
571 C      THE FOLLOWING SUBROUTINE CALCULATES THE ANGULAR FREQUENCY FOR
572 C      AN INTERVAL
573 C
574 C
575 C      SUBROUTINE FREQ(X0,X1,N,PSTAR)
576 C      IMPLICIT REAL=8 (A-M,0-Z)
577 C      REAL=S N
578 C      COMMON/Coeff/A,B
579 C      FORA=X0*B/(X0**2)
580 C      P1=A*X1*B/(X1**2)
581 C      SLOPE=(P0-P1)/(X0-X1)
582 C      PSTAR=DSQRT(SLOPE-N**2)
583 C
584 C      RETURN
585 C
586 C
587 C      THE FOLLOWING SUBROUTINE CALCULATES THE TIME INTERVAL FOR A
588 C      PARTICULAR SEGMENT BY USING A NEWTON-RAPHSON METHOD. IT ALSO HAD
589 C      A PROVISION FOR FINDING THE LAST TIME INTERVAL PRIOR TO THE FIRST
590 C      TROUGH ON THE DISPLACEMENT-TIME CURVE.
591 C
592 C
593 C      SUBROUTINE TIME(X0,XDOTO,X1,DELT,XSTAR,N,P,DELTAT)
594 C      IMPLICIT REAL=8 (A-M,0-Z)
595 C      REAL=S N
596 C      DELTA=DELT+0.08
597 C      C1=X0-XSTAR
598 C      C2=(N*(X0-XSTAR)+XDOTO)/P
599 C      C3=N*C1+P*C2
600 C      C4=P*C1-N*C2
601 C      C5=(P**2-N**2)+C1**2.0=P**2-N**2
602 C      C6=C2*(P**2-N**2)-C1**2.0=P**2-N**2
603 C
604 C      PMAIN FUNCTION (DISPLACEMENT)
605 C      FP=F-PRIME (VELOCITY)
606 C
607 C
608 C      PFP=F-PRIME (ACCELERATION)
609 C
610 C
611 C      700 CONTINUE
612 C      PX1=XSTAR-DEXP(-N*DELT)=(C1*DCOS(P*DELT))+C2*DSIN(P*DELT)
613 C      PFP=-DEXP(-N*DELT)*(C3*DCOS(P*DELT))+C4*DSIN(P*DELT)
614 C      IF(PFP.LT.-1.0D-15) GO TO 710
615 C      DELTA1=DELT-F/FP
616 C      DIFF=DELTA1-DELT
617 C      IF(DABS(DIFF).LT.1.0D-08) GO TO 730
618 C      DELTA=DELTA1
619 C      GO TO 700
620 C
621 C
622 C      AS THE DISPLACEMENT APPROACHES A PEAK OR TROUGH, THE VELOCITY
623 C      APPROACHES ZERO. THE RESULT IS THAT INSTEAD OF F AND FP, THE
624 C      FOLLOWING ARE USED IN THE RESPECTIVE POSITIONS:
625 C
626 C      PFP=F-PRIME (VELOCITY)
627 C      FPFP=FP-PRIME (ACCELERATION)
628 C
629 C      AND THE CALCULATION CONTINUES.
630 C
631 C
632 C      710 CONTINUE
633 C      DELTA=DELT+0.08
634 C      720 CONTINUE
635 C      PFP=-DEXP(-N*DELT)=(C3*DCOS(P*DELT))+C4*DSIN(P*DELT)
636 C      FPFP=-DEXP(-N*DELT)*(C5*DCOS(P*DELT))+C6*DSIN(P*DELT)
637 C      DELTA1=DELT-FP/FPFP
638 C      DIFF=DELTA1-DELT
639 C      IF(DABS(DIFF).LT.1.0D-08) GO TO 730
640 C      DELTA=DELTA1
641 C      GO TO 720
642 C
643 C      730 CONTINUE
644 C      DELTAT=DELT
645 C      RETURN
646 C
647 C
648 C      SUBROUTINE SPEED(X0,XDOTO,XSTAR,N,DELTAT,P,XDOT1)
649 C      IMPLICIT REAL=8 (A-M,0-Z)
650 C      REAL=S N
651 C      C1=XDOTO
652 C      C2=-N*(X0-XSTAR)+XDOTO/P-P*(X0-XSTAR)
653 C      XDOT1=DEXP(-N*DELTAT)*(C1*DCOS(P*DELTAT)+C2*DSIN(P*DELTAT))
654 C
655 C
656 C
657 C
658 C
659 C
660 C

```

END OF FILE

POOR COPY
COPIE DE QUALITEE INFERIEURE

```

1  C*****
2  C
3  C  PROGRAM HSPRING0
4  C
5  C
6  C
7  C  THE FOLLOWING PROGRAM CALCULATES THE SOLUTION FOR THE
8  C  EQUATION:
9  C
10 C  X(DOUBLE-DOT)+Q=(ABS(X(DOT))**2)*SGN(X(DOT))+A*X+B*(X+3)*C
11 C
12 C  OBTAINED BY PIECEWISE LINEARIZATION USING CHORDS, AS WELL AS
13 C  FINDING THE FIRST TROUGH ON THE DISPLACEMENT-TIME CURVE.
14 C
15 C
16 C  THE RESULTS ARE FOUND FROM THE FOLLOWING:
17 C
18 C  X=XSTAR*EXP(-N*(T-T0))=(C1*COS(PSTAR*(T-T0))
19 C  +C2*SIN(PSTAR*(T-T0)))
20 C
21 C  X(DOT)=EXP(-N*(T-T0))*(C3*COS(PSTAR*(T-T0))
22 C  +C4*SIN(PSTAR*(T-T0)))
23 C
24 C  C1=X0-XSTAR
25 C  C2=(N*(X0-XSTAR)+XDOT0)/P
26 C  C3=N*C1*PEC2
27 C  C4=P*C1-C2
28 C
29 C  THE VARIOUS PARAMETERS ARE EXPLAINED AND CALCULATED IN THE
30 C  PROGRAM.
31 C
32 C
33 C  THE PROGRAM ALSO SOLVES THE EQUATION USING FIFTH- AND SIXTH-
34 C  ORDER RUNGE-KUTTA METHODS. BOTH SOLUTIONS ARE PLOTTED.
35 C
36 C*****
```

5

```

40
41
42
43
44
45
46
47  IMPLICIT REAL*8 (A-H,O-Z)
48  EXTERNAL FCN
49  REAL*8 Y(2),CN(24),W(2,20)
50  REAL*8 N
51  REAL*8 AMPLP(200),AMPLRK(200),TP(200),TRK(200)
52  REAL*8 YP(2000),TPP(200)
53  REAL*4 XA(2),YA(2)
54  REAL*4 HA,HG,HC,VA,V,V
55  INTEGER*4 ALPH(20)
56  COMMON/COEFF/A,B
57  COMMON/DAMP/Q
58
59  C
60  C  THE FOLLOWING VALUES ARE READ FROM DATA FILE PFITODATA:
61  C
62  C
63  C  A,B,Q AS IN THE EQUATION
64  C
65  C  DISPL0=INITIAL AMPLITUDE
66  C  V0=INITIAL VELOCITY
67  C  TIME0=INITIAL TIME
68  C
69  C  N=NUMBER OF DIVISIONS OF INITIAL DISPLACEMENT
70  C  DTIME=TIME INCREMENT (RUNGE-KUTTA SOLUTION)
71  C
72  C  HA,HG=PLOT PARAMETERS FOR HORIZONTAL AXIS (HG AN INTEGRAL MULTIPLE
73  C  OF 7.0)
74  C
75  C  VC=PLOT PARAMETER FOR VERTICAL AXIS (AN INTEGRAL MULTIPLE OF 12.0,
76  C  12.0 GIVING BEST RESULTS)
77  C
78  C  XA,YA=PARAMETERS FOR PLOTTING ZERO LINE
79  C
80  C  ALPH=ARRAY FOR PLOT AXIS LABELS
81
82  C
83  READ(5,10) A,B,Q
84  10 FORMAT(2D10.6)
85  READ(5,20) DISPL0,V0,TIME0
86  20 FORMAT(2D10.6)
87  READ(5,30) H,DTIME
88  30 FORMAT(2D10.6)
89  READ(5,40) HA,HG
90  40 FORMAT(2F10.4)
91  READ(5,50) VC
92  50 FORMAT(F10.4)
93  READ(5,60) XA(1)
94  60 FORMAT(F10.4)
95  READ(5,70) YA(1),YA(2)
96  70 FORMAT(2F10.4)
97  READ(5,80) (ALPH(I),I=1,12)
98  80 FORMAT(12A4)
99  READ(5,90) (ALPH(I),I=13,18)
100 90 FORMAT(4A4)
101 READ(5,100) (ALPH(I),I=19,20)
102 100 FORMAT(4A4)
103
104  C
105  C=DAMPING COEFFICIENT (INITIALLY SET AT ZERO)
106  C
107  C  N=DAMPING FACTOR
108  C
109  C=0.0
110  H=C/2.0

```

```

111 C#####
112 C      SETTING THE INITIAL CONDITIONS.
113 C
114 C      X0=INITIAL DISPLACEMENT FOR EACH SUCCESSIVE SEGMENT
115 C      XDDOT=INITIAL VELOCITY FOR EACH SEGMENT
116 C      XDDOTP=PREVIOUS FINAL SEGMENT VELOCITY
117 C      T0=CUMULATIVE TIME TAKEN TO REACH BEGINNING OF SEGMENT
118 C      DELT=INITIAL GUESS AT TIME INTERVAL FOR THE SEGMENT BEING EXAMINED
119 C          (SEE SUBROUTINE TIME BELOW)
120 C
121 C      DELTAX=LENGTH OF SEGMENT
122 C
123 C      I=COUNTER FOR DATA POINTS TO BE PLOTTED
124 C
125 C#####
126 C      X0=DISPLO
127 C      XDDOT0=VO
128 C      XDDOTP=VO
129 C      T0=TIME0
130 C      DELT=TIME0
131 C      DELTAX=DISPLO/H
132 C      DELX=DELTAX
133 C      I=0
134 C
135 C#####
136 C      THE EQUATION PARAMETERS ARE WRITTEN INTO OUTPUT FILE PFITONUM.
137 C
138 C
139 C#####
140 C      WRITE(10,110)
141 C      110 FORMAT('1','PIECEWISE LINEAR PERIOD FOR THE EQUATION:')
142 C      WRITE(10,120)
143 C      120 FORMAT('0','(X(DOUBLE-DOT)+0*(ABS(X(DOT))**2)=SGN(X(DDOT))+A*X+B*X**'
144 C          '13)=0')
145 C      WRITE(10,130) A,B,C
146 C      130 FORMAT('0','A= ',G20.10,2X,'B= ',G20.10,2X,'C= ',G20.10)
147 C      WRITE(10,140) DISPLO,VO
148 C      140 FORMAT('0','X(0)= ',G20.10,2X,'X(DOT)(0)= ',G20.10)
149 C      WRITE(10,150)
150 C      150 FORMAT('0','X, 'AMPLITUDE', 10X, 'CUMULATIVE TIME')
151 C
152 C      X1=DISPLACEMENT AT END OF SEGMENT
153 C
154 C
155 C#####
156 C      X1=X0+DELTAX
157 C      160 CONTINUE
158 C      I=I+1
159 C      170 CONTINUE.
160 C
161 C      XPRIME=SUBROUTINE FOR CALCULATING POINT WHERE SEGMENT CHORD LINE
162 C      CROSSES X-AXIS
163 C      XSTAR=CALCULATED VALUE
164 C
165 C      FREQ=SUBROUTINE FOR CALCULATING ANGULAR FREQUENCY
166 C      PSTAR=CALCULATED VALUE
167 C
168 C      TIME=SUBROUTINE FOR CALCULATING TIME INTERVAL FOR A SEGMENT
169 C      DELTAT=CALCULATED VALUE
170 C
171 C      SPEED=SUBROUTINE FOR CALCULATING SPEED AT END OF SEGMENT
172 C      XDOT1=CALCULATED VALUE
173 C
174 C      **NOTE** THE PROGRAM STOPS IF THE DISPLACEMENT AT THE END OF A SEGMENT
175 C      IS EQUAL TO -DISPLO (I. E., UNDAMPED CASE).
176 C
177 C#####
178 C      CALL XPRIME(X0,X1,XSTAR)
179 C      CALL FREQ(X0,X1,N,PSTAR)
180 C      CALL TIME(X0,XDDOT,X1,DELT,XSTAR,N,PSTAR,DELTAT)
181 C      CALL SPEED(X0,XDDOT,XSTAR,N,DELTAT,PSTAR,XDOT1)
182 C
183 C
184 C      THE FOLLOWING IS THE ITERATION TO FIND THE END VELOCITY FOR THE
185 C      PRESENT SEGMENT.
186 C
187 C      DELTAV=DIFFERENCE BETWEEN XDOT1 AND XDDOTP
188 C      XDOTAV=AVERAGE VELOCITY FOR SEGMENT
189 C      SGNYAV=SGN(XDOTAV)
190 C
191 C      CNEW DAMPING COEFFICIENT FOR NEXT ITERATION
192 C
193 C#####
194 C      DELTAV=XDOT1-XDDOTP
195 C      IF(DABS(DELTAV).LT.1.0D-05) GO TO 180
196 C      XDDOTP=XDOT1
197 C      XDOTAV=(XDOT1+XDDOTP)/2.0
198 C      SGNYAV=XDOTAV/DABS(XDOTAV)
199 C      C0=(DABS(XDOTAV))**2=SGNYAV/XDOTAV
200 C      H=C/2.0
201 C
202 C      GO TO 170
203 C      180 CONTINUE
204 C
205 C      STARTING AT STEP 210 AND CONTINUING AT 220, THE PROGRAM TESTS FOR
206 C      ZERO VELOCITY. IF IT OCCURS AT THE END OF THE SEGMENT, THE
207 C      PROGRAM STOPS (SEE STEP 320). IF NOT, THE LOCATION OF THE FIRST
208 C      TROUGH IS SOUGHT (STEPS 270 TO 280).
209 C
210 C
211 C      DELT=DELTAT
212 C      T1=DELTAT
213 C      IF(X0.GE.0.0) GO TO 180
214 C      GO TO 210
215 C
216 C      180 CONTINUE
217 C
218 C      AMPLP=ARRAY FOR X0
219 C      TP=ARRAY FOR T0
220 C

```

```

222 C
223 C#####AMPLP(I)=X0
224 C#####TP(I)=TO
225 C
226 C THE INTERIM VALUES ARE WRITTEN INTO OUTPUT FILE PFITONUM.
227 C
228 C#####WRITE(10,200) X0,TO
229 C#####200 FORMAT(1X,2(G20.10))
230 C
231 C#####RESETTING THE INITIAL CONDITIONS.
232 C
233 C#####X0=X1
234 C#####X1=X0-DELTAX
235 C#####X0DT0=X0DT1
236 C#####T0T1
237 C#####GO TO 180
238 C#####110 CONTINUE
239 C#####IF(DABS(X0DT1).LT.1.0D-08) GO TO 220
240 C#####GO TO 180
241 C#####220 CONTINUE
242 C
243 C THE SEGMENT IN WHICH THE FIRST TROUGH OCCURS IS WRITTEN INTO
244 C OUTPUT FILE PFITONUM.
245 C
246 C#####AMPLP(I)=X0
247 C#####TP(I)=TO
248 C#####WRITE(10,230) X0,TO
249 C#####230 FORMAT(1X,2(G20.10))
250 C#####DIFFP=DISPL0-X0
251 C#####IF(DABS(DIFFP).LT.1.0D-08) GO TO 320
252 C#####WRITE(10,240) X1,T1
253 C#####240 FORMAT(1X,2(G20.10))
254 C#####WRITE(10,250)
255 C#####250 FORMAT(' ','TROUGH ITERATIONS')
256 C#####WRITE(10,250)
257 C#####250 FORMAT('0','10X,X0','20X,X1','14X,X1(CALCULATED)')
258 C
259 C THE FOLLOWING (ENDING AT STEP 280) IS THE CALCULATION OF THE FIRST
260 C TROUGH. USING THE TIME INTERVAL REQUIRED TO REACH ZERO VELOCITY,
261 C A NEW VALUE OF X1 (XF) IS CALCULATED AND COMPARED TO THE PREVIOUS
262 C VALUE. THIS NEW VALUE BECOMES THE NEW X1 IF THE DIFFERENCE
263 C BETWEEN XF AND THE OLD X1 IS NOT WITHIN A SET TOLERANCE, AND THE
264 C PROCESS STARTS AGAIN.
265 C
266 C THESE ITERATIONS ARE WRITTEN INTO OUTPUT FILE PFITONUM.
267 C
268 C#####270 CONTINUE
269 C#####C1=X0-XSTAR
270 C#####C2=(X0-XSTAR)+X0DT0/PSTAR
271 C#####P=PSTAR
272 C#####XP=XSTAR+DEXP(-N*DELTAT)*(C1=DCOS(P*DELTAT)+C2=DSIN(P*DELTAT))
273 C#####WRITE(10,280) X0,X1,XP
274 C#####280 FORMAT(2(F12.8,2X),2X,F12.8)
275 C#####DXFP=X1-XF
276 C#####IF(DABS(DIFFP).LT.1.0D-08) GO TO 290
277 C#####X1=XP
278 C#####DIFFP=X0-XF
279 C#####IF(DABS(DIFFP).LT.1.0D-08) GO TO 280
280 C#####CALL XPRIME(X0,X1,XSTAR)
281 C#####CALL FREQ(X0,X1,N,PSTAR)
282 C#####CALL TIME(X0,X0DT0,X1,DELT,XSTAR,N,PSTAR,DELTAT0)
283 C#####GO TO 270
284 C#####290 CONTINUE
285 C
286 C THE FINAL VALUES ARE WRITTEN INTO OUTPUT FILE HSPRINGONUM.
287 C
288 C#####TF=T1
289 C#####XP=X1
290 C#####I=I+1
291 C#####NPOINT=I
292 C#####AMPLP(I)=XF
293 C#####TP(I)=TF
294 C#####TAU=2.0*T1
295 C#####WRITE(10,300) DELTAX,TAU
296 C#####300 FORMAT(' ','DELTAX=',G20.10,2X,'PERIOD=',G20.10)
297 C#####WRITE(10,310) XP
298 C#####310 FORMAT('0','FIRST TROUGH OCCURS AT X*',G20.10)
299 C#####GO TO 340
300 C#####320 CONTINUE
301 C
302 C THE FINAL VALUES ARE WRITTEN INTO OUTPUT FILE HSPRINGONUM (UNDAMPED
303 C CASE).
304 C
305 C#####NPOINT=I
306 C#####TAU=2.0*T1
307 C#####WRITE(10,330) DELTAX,TAU
308 C#####330 FORMAT(' ','DELTAX=',G20.10,2X,'PERIOD=',G20.10)
309 C#####340 CONTINUE
310 C
311 C
312 C THE FOLLOWING SECTION (ENDING AT LINE 380) IS THE RUNGE-KUTTA
313 C SOLUTION OF THE SAME ORIGINAL EQUATION.
314 C
315 C#####TFINAL=FINAL-TIME
316 C#####INTVL=NUMBER OF INCREMENTS

```

```

331      C
332      C#####TFINAL=0.1
333      C#####INTVL=TFINAL/DTIME+1
334      C#####C
335      C##### THE FOLLOWING ARE PARAMETERS FOR THE NUMERICAL SUBROUTINE
336      C##### CONSULT THE IMSL MANUAL ON "OVERK" FOR FURTHER DETAILS.
337      C
338      C#####C
339      C##### XRK=TIME0
340      C##### Y(1)=DISPL0
341      C##### Y(2)=VO
342      C##### NEON=2
343      C##### NW=2
344      C##### TOL=1.OD-12
345      C##### IND=1
346      C##### XEND=DTIME
347      C#####C
348      C##### ZEIT=CUMULATIVE VALUE OF TIME AT END OF EACH INCREMENT
349      C#####C
350      C##### ZEIT=0.0
351      C#####C
352      C##### AMPLRK=ARRAY FOR AMPLITUDE VALUES
353      C##### TRK=ARRAY FOR TIME VALUES
354      C#####C
355      DO 350 J=1,INTVL
356      AMPLRK(J)=Y(1)
357      TRK(J)=ZEIT
358      CALL OVERK(NEON,FCH,XRK,XEND,TOL,IND,CH,NW,W,IER)
359      C#####C
360      C##### INCREMENTING THE TIME VALUES.
361      C#####C
362      ZEIT=XEND
363      XEND=XEND+DTIME
364      350 CONTINUE.
365      C#####C
366      C##### THE CALCULATED VALUES ARE WRITTEN INTO OUTPUT FILE HSPRINGORK.
367      C#####C
368      WRITE(11,360)
369      360 FORMAT('1',RUNGE-KUTTA RESULTS FOR THE EQUATION:')
370      WRITE(11,370)
371      FORMAT('0', '(DOUBLE-DOT)+C=X(DOT)+B*X+B*(X**3)*O')
372      WRITE(11,380) A,B,C
373      380 FORMAT('0', 'A', G20.10, 2X, 'B', G20.10, 2X, 'C', G20.10)
374      WRITE(11,380) DISPL0, VO
375      380 FORMAT('0', 'X(0)', G20.10, 2X, 'X(DOT)(0)', G20.10)
376      WRITE(11,400)
377      400 FORMAT('1', 'X, 'TIME', 18X, 'AMPLITUDE')
378      DO 420 NO=1,INTVL
379      WRITE(11,410) TRK(NO),AMPLRK(NO)
380      410 FORMAT(1X,2(G20.10,2X))
381      420 CONTINUE
382      C#####C
383      C##### THE NEXT FEW LINES ARE USED IN FINALIZING ALL THE PLOT PARAMETERS
384      C##### WHICH INCLUDES AXIS SCALING.
385      C#####C
386      M=1
387      IF(DISPL0.LE.1.0) GO TO 440
388      430 CONTINUE
389      DECD=DISPL0/(10.0**M)
390      IF(DECD.LE.1.0) GO TO 460
391      M=M+1
392      GO TO 430
393      440 CONTINUE
394      DIGDISPL0=(10.0**M)
395      IF(DIGD.GE.1.0) GO TO 480
396      M=M+1
397      GO TO 440
398      450 CONTINUE
399      DECD=DIGD/10.0
400      M=(M-1)
401      460 CONTINUE
402      IF(DECD.LE.0.1) VFACT=G1
403      IF((DECD.GT.0.1).AND.(DECD.LE.0.25)) VFACT=G0.25
404      IF((DECD.GT.0.25).AND.(DECD.LE.0.5)) VFACT=G0.5
405      IF((DECD.GT.0.5).AND.(DECD.LE.1.0)) VFACT=G1.0
406      SFACT=VC/10.0
407      VAR=VFACT*SFACT=10.0**M
408      DEN=VC/(2.0*SFACT)
409      VBR=(VFACT/DEN)=10.0**M
410
411      M=1
412      IF(TFINAL.LE.1.0) GO TO 480
413      470 CONTINUE
414      DECT=TFINAL/(10.0**M)
415      IF(DECT.LE.1.0) GO TO 500
416      M=M+1
417      GO TO 470
418      480 CONTINUE
419      DIGTTFINAL=(10.0**M)
420      IF(DIGT.GE.1.0) GO TO 490
421      M=M+1
422      GO TO 480
423      490 CONTINUE
424      DECT=DECT/10.0
425      M=(M-1)
426      500 CONTINUE
427      IF(DECT.LE.0.07) HFACT=G0.1
428      IF((DECT.GT.0.07).AND.(DECT.LE.0.14)) HFACT=G0.2
429

```

```

441      IF((DECT,ST,0.14), AND, (DECT,LE,0.35)) HFACT=0.5
442      IF((DECT,ST,0.35), AND, (DECT,LE,0.7)) HFACT=1.0
443      IF((DECT,ST,0.7), AND, (DECT,LE,1.0)) HFACT=2.0
444      HBNFACT=10.0*(H-1)
445      LONGHC
446      XA(2)=DFLOAT(LONG)+HB-0.005
447
448      C THE FOLLOWING SECTION SETS UP THE PIECEWISE LINEAR RESULTS SO THAT
449      C NO MORE THAN 12 POINTS ARE PLOTTED FOR A CLEARER PRESENTATION.
450
451      C YP=ARRAY FOR PLOTTED PIECEWISE LINEAR DISPLACEMENTS
452      C TPP=ARRAY FOR PLOTTED PIECEWISE LINEAR TIME VALUES.
453
454      COUNT=DFLOAT(NPOINT)
455      INC=COUNT/10.0
456      DELINC=COUNT/10.0-DFLOAT(INC)
457      IF(DELINC<=0.5) INC=INC+1
458
459      NPA1
460
461      NDA1
462      S10 CONTINUE
463      YP(NP)=AMPLP(ND)
464      TPP(NP)=TP(ND)
465      NPA(NP)=1
466      ND=ND+INC
467      IF(ND.GE.POINT) GO TO S20
468      GO TO S10
469
470      S20 CONTINUE
471      YP(NP)=AMPLP(NPOINT)
472      TPP(NP)=TP(NPOINT)
473
474      C PLOTTING THE SOLUTIONS. CONSULT THE WRITEUP ON CGPL/CGPL2 AND
475      C THE MANUAL ON DIGITAL PLOTTING FOR DETAILS.
476
477      C
478      ND=INTYL
479      NF=1
480      CALL CGPL2(TRK,AMPLRK,ND,NF,S,HA,HB,HC,VA,VB,VC,ALPH)
481      NO=NPF
482      CALL CGPL2(TPP,YP,ND,2,1,HA,HB,HC,VA,VB,VC,ALPH)
483      NF=4
484      CALL CGPL2(XA,YA,2,NF,4,HA,HB,HC,VA,VB,VC,ALPH)
485      STARTX=(HC-8.3)/2.0
486      STARTY=VC+0.5
487      HORIZ=HA
488      VERT=VC+0.5
489      CALL PLOT(HORIZ,VERT,3)
490      HORIZ=HC
491      CALL PLOT(HORIZ,VERT,2)
492      VERT=VERT+4.8
493      CALL PLOT(HORIZ,VERT,2)
494      HORIZ=HA
495      CALL PLOT(HORIZ,VERT,2)
496      VERT=VC+0.5
497      CALL PLOT(HORIZ,VERT,2)
498      HORIZ=STARTX+0.2
499      VERT=STARTY+4.2
500      CALL SYMBOL(HORIZ,VERT,0.2,'A'=-,0,0,3)
501      HORIZ=STARTX+1.0
502      CALL NUMBER(HORIZ,VERT,0.2,A,0,0,4)
503      HORIZ=STARTX+3.0
504      CALL SYMBOL(HORIZ,VERT,0.2,'B'=-,0,0,3)
505      HORIZ=STARTX+3.8
506      CALL NUMBER(HORIZ,VERT,0.2,B,0,0,4)
507      HORIZ=STARTX+2.0
508      VERT=STARTY+3.8
509      CALL SYMBOL(HORIZ,VERT,0.2,'C'=-,0,0,3)
510      HORIZ=STARTX+2.8
511      CALL NUMBER(HORIZ,VERT,0.2,C,0,0,4)
512      HORIZ=STARTX+0.2
513      VERT=STARTY+3.1
514      CALL SYMBOL(HORIZ,VERT,0.2,E2,0.0,-1)
515      HORIZ=STARTX+0.4
516      CALL SYMBOL(HORIZ,VERT,0.2,'D'=-,0,0,3)
517      HORIZ=STARTX+1.0
518      VERT=STARTY+2.8
519      CALL SYMBOL(HORIZ,VERT,0.2,'LINEARIZATION'=-,0,0,18)
520      HORIZ=STARTX+8.4
521      CALL NUMBER(HORIZ,VERT,0.2,DELX,0,0,4)
522      HORIZ=STARTX+0.2
523      VERT=STARTY+2.4
524      CALL SYMBOL(HORIZ,VERT,0.2,E2,0.0,-1)
525      HORIZ=STARTX+0.4
526      CALL SYMBOL(HORIZ,VERT,0.2,'T (RUNGE-KUTTA)'=-,0,0,17)
527      HORIZ=STARTX+4.0
528      CALL NUMBER(HORIZ,VERT,0.2,DTIME,0,0,4)
529      HORIZ=STARTX+0.2
530      VERT=STARTY+1.8
531      CALL SYMBOL(HORIZ,VERT,0.2,'X(0)'=-,0,0,8)
532      HORIZ=STARTX+1.8
533      CALL NUMBER(HORIZ,VERT,0.2,DISPL0,0,0,4)
534      HORIZ=STARTX+3.5
535      CALL SYMBOL(HORIZ,VERT,0.2,'X(0)'=-,0,0,8)
536      HORIZ=STARTX+3.528
537      VERT=STARTY+1.8
538      CALL SYMBOL(HORIZ,VERT,0.2,75,0,0,-1)
539      HORIZ=STARTX+4.8
540      VERT=STARTY+1.8
541      CALL NUMBER(HORIZ,VERT,0.2,V0,0,0,4)
542      HORIZ=STARTX+0.3
543      VERT=STARTY+0.7
544      CALL PLOT(HORIZ,VERT,3)
545      HORIZ=STARTX+0.8
546      CALL PLOT(HORIZ,VERT,2)
547      HORIZ=STARTX+1.0
548      VERT=STARTY+0.5
549      CALL SYMBOL(HORIZ,VERT,0.2,'RUNGE-KUTTA',0,0,11)
550      HORIZ=STARTX+0.8

```

POOR COPY
COPIE DE QUALITEE INFERIEURE

POOR COPY
COPIE DE QUALITEE INFERIEURE

```

861      DELTA1=DELTA-PP/PPP
862      DPP=DELTA1-DELTA
863      IF(DABS(DPP).LT.1.0D-08) GO TO 730
864      DELTA=DELTA1
865      GO TO 720
866      730 DELTA1=DELTA
867      RETURN
868      END.
869
870      C
871      C
872      C***** THE FOLLOWING SUBROUTINE CALCULATES THE VELOCITY AT THE END OF THE
873      C***** SEGMENT BEING EXAMINED.
874
875      C***** SUBROUTINE SPEED(XO,XDOTO,XSTAR,N,DELTAT,P,XDOT1)
876      IMPLICIT REAL*8 (A-H,D-Z)
877      REAL*8 H
878      C1=XDOT0
879      C2=N*(XO-XSTAR)+XDOTO/P-PP*(XO-XSTAR)
880      XDOT1=DEXP(-H*DELTAT)*(C1+DCOS(P*DELTAT))+C2+DSIN(P*DELTAT)
881      RETURN
882      END
883
884
885      C
886
887      C***** THE FOLLOWING SUBROUTINE CONTAINS THE DIFFERENTIAL EQUATIONS TO BE
888      C***** SOLVED BY OVERK: THE INSTANTANEOUS VELOCITY AND THE ORIGINAL
889      C***** EQUATION. "DFORCE" IS THE DAMPING FORCE.
890
891      C***** SUBROUTINE FCM(N,X,Y,YPRIME)
892      IMPLICIT REAL*8 (A-H,D-Z)
893      REAL*8 Y(N),YPRIME(N),X
894      COMMON/COEFF/A,B
895      COMMON/DAMP/C
896      YPRIME(1)=Y(2)
897      IF(DABS(YPRIME(1)).LT.1.0D-08) GO TO 800
898      SEN=YPRIME(1)/DABS(YPRIME(1))
899      GO TO 810
900      800 CONTINUE
901      SENY=DO
902      810 CONTINUE
903      DFORCE=0*(DABS(YPRIME(1))**2)*SENY
904      YPRIME(2)=-(DFORCE+A*Y(1)+B*(Y(1)**3))
905      RETURN
906      END
907
908      END OF FILE

```

```

1 C*****
2 C
3 C PROGRAM VANDERPOL
4 C
5 C
6 C
7 C THE FOLLOWING PROGRAM CALCULATES THE SOLUTION FOR THE VAN DER
8 C POL EQUATION:
9 C
10 C X(DOUBLE-DOT)+MU*(X**2-1)*X(DOT)*X=0
11 C
12 C OBTAINED BY PIECEWISE LINEARIZATION USING CHORDS OVER SEVERAL
13 C CYCLES OF THE DISPLACEMENT-TIME CURVE.
14 C
15 C
16 C THE RESULTS ARE BASED ON THE STANDARD FORM:
17 C
18 C X=C*EXP(R*DELTAT)
19 C
20 C R1=N+SORT(N==2-P==2)
21 C R2=N-SORT(N==2-P==2)
22 C
23 C WHERE THE UNDERDAMPED, CRITICALLY DAMPED, AND OVERDAMPED
24 C APPROXIMATIONS WOULD DEPEND UPON MU AND AN ESTIMATED
25 C DISPLACEMENT.
26 C
27 C THE VARIOUS PARAMETERS ARE EXPLAINED AND CALCULATED IN THE
28 C PROGRAM.
29 C
30 C
31 C THE PROGRAM ALSO SOLVES THE EQUATION USING FIFTH AND SIXTH-
32 C ORDER RUNGE-KUTTA METHODS. BOTH SOLUTIONS ARE PLOTTED, AS
33 C WELL AS THE PHASE-PLANE DIAGRAMS.
34 C*****
35 C
36 C
37 C
38 C
39 C
40 C
41 C
42 C
43 C
44 C
45 C
46 IMPLICIT REAL*8 (A-H,D-Z)
47 EXTERNAL FCN
48 REAL*8 Y(2),CM(24),W(2,20)
49 REAL*8 K,MU,N
50 REAL*4 AMPLP(10000),VP(10000),TP(10000)
51 REAL*4 AMPLRK(10000),YRK(10000),TRK(10000)
52 REAL*4 YP(10000),VPP(10000),TPP(10000)
53 REAL*4 YRK(10000),VRK(10000),TPRK(10000)
54 REAL*4 HAD,HBD,HCD,VAD,VBD,VCD
55 REAL*4 HAP,HEP,HCP,VAP,VBP,VCP
56 REAL*4 XAD(2),YAD(2),XAPH(2),YAPH(2),XAPV(2),YAV(2)
57 INTEGER*4 ALPHD(20),ALPHP(20)
58 COMMON MU
59 C#####
60 C
61 C THE FOLLOWING VALUES ARE READ FROM DATA FILE VDPFDATA:
62 C
63 C
64 C MUS IS IN THE EQUATION
65 C
66 C DISPLAY=INITIAL AMPLITUDE
67 C VO=INITIAL VELOCITY
68 C TIME0=INITIAL TIME
69 C
70 C DELTAT=TIME INCREMENT (PIECEWISE SOLUTION)
71 C NCYCLES=NUMBER OF HALF-CYCLES (PIECEWISE SOLUTION)
72 C DT=TIME INCREMENT (RUNGE-KUTTA SOLUTION)
73 C
74 C HAD,HCD=PLOT PARAMETERS FOR HORIZONTAL AXIS OF DISPLACEMENT-TIME
75 C CURVE (HCD AN INTEGRAL MULTIPLE OF 7.0)
76 C
77 C VCD=PLOT PARAMETER FOR VERTICAL AXIS OF DISPLACEMENT-TIME CURVE
78 C (AN INTEGRAL MULTIPLE OF 12.0, 12.0 GIVING BEST RESULTS)
79 C
80 C XAD,YAD=PLOT PARAMETERS FOR PLOTTING ZERO LINE OF DISPLACEMENT-
81 C TIME CURVE.
82 C
83 C ALPHD=ARRAY FOR PLOT AXIS LABELS OF DISPLACEMENT-TIME CURVE
84 C
85 C VEP=PLOT PARAMETER FOR VERTICAL AXIS OF PHASE-PLANE DIAGRAM (AN
86 C INTEGRAL MULTIPLE OF 12.0, 12.0 GIVING BEST RESULTS)
87 C
88 C ALPHP=ARRAY FOR PLOT AXIS LABELS OF PHASE-PLANE DIAGRAM
89 C
90 C#####
91 C
92 READ(5,10) MU
93 10 FORMAT(D10.8)
94 READ(5,20) DISPLAY,VO,TIME0
95 20 FORMAT(3D10.8)
96 READ(5,30) DELTAT,NCYCLES,DT
97 30 FORMAT(3D10.8)
98 READ(5,40) HAD,HCD
99 40 FORMAT(2F10.4)
100 READ(5,50) VCD
101 50 FORMAT(F10.4)
102 READ(5,60) XAD(1)
103 60 FORMAT(F10.4)
104 READ(5,70) YAD(1),YAD(2)
105 70 FORMAT(2F10.4)
106 READ(5,80) (ALPHD(I),I=1,12)
107 80 FORMAT(12A4)
108 READ(5,90) (ALPHD(I),I=13,16)
109 90 FORMAT(4A4)
110 READ(5,100) (ALPHD(I),I=17,20)
110 100 FORMAT(4A4)

```

```

111      READ(5,110) VCP
112      110 FORMAT(F10.4)
113      READ(5,120) (ALPHP(I),I=1,12)
114      120 FORMAT(12A4)
115      READ(5,130) (ALPHP(I),I=13,18)
116      130 FORMAT(4A4)
117      READ(5,140) (ALPHP(I),I=17,20)
118      140 FORMAT(4A4)
119
120      C
121      C      SETTING THE INITIAL CONDITIONS.
122      C
123      C      X0=INITIAL DISPLACEMENT FOR EACH SUCCESSIVE SEGMENT
124      C      XD0T0=INITIAL VELOCITY FOR EACH SEGMENT
125      C      T0=INITIAL TIME FOR EACH SEGMENT
126      C      XPREV=PREVIOUS FINAL SEGMENT DISPLACEMENT
127
128      C#####
129      X0=DISPL0
130      XD0T0=VO
131      T0=TIME0
132      XPREV=DISPL0
133
134      C
135      C      SIGSUBROUTINE FOR CALCULATING SGNX(X)
136      C      SGNX=CALCULATED VALUE
137
138      C#####
139      CALL SIG(X0,SGNX0)
140
141      C
142      I=COUNTER USED FOR PLOTTING PURPOSES
143
144      K=KOUNTER FOR THE NUMBER OF HALF-CYCLES (AN INTEGRAL MULTIPLE OF
145      10.0, 10.0 GIVING BEST RESULTS)
146
147
148      I=0
149      K=1.0
150
151
152      C      THE EQUATION PARAMETERS ARE WRITTEN INTO OUTPUT FILE VOPPNUM.
153
154      C#####
155      WRITE(10,150)
156      150 FORMAT(1X,'PIECEWISE LINEAR RESULTS FOR THE EQUATION: ')
157      WRITE(10,160)
158      160 FORMAT('0', 'X(DOUBLE-DOT)+MU*(X**2-1)*X(DOT)+X0')
159      WRITE(10,170) MU
160      170 FORMAT('0', 'MU', G20.10)
161      WRITE(10,180) DISPL0,VO
162      180 FORMAT('0', 'X(0)=', G20.10,X,'X(DOT)(0)=', G20.10)
163      WRITE(10,180)
164      180 FORMAT('0', 'X TIME', 18X,'X',18X,'VELOCITY')
165
166
167      C      THE FOLLOWING IS THE CALCULATION OF THE SOLUTION FOR A PARTICULAR
168      C      HALF-CYCLE, STARTING AT STEP 200 AND ENDING AT STEP 260.
169
170      C      XDISPLACEMENT USED FOR CALCULATING A DAMPING COEFFICIENT (SEE
171      C      BELOW)
172      C      TINCUMULATIVE TIME.
173
174
175      200 CONTINUE
176      I=I+1
177      X=X0
178      T=T0+DELTAT
179
180
181      C      THE FOLLOWING IS THE ITERATION TO FIND THE END DISPLACEMENT FOR
182      C      THE PRESENT SEGMENT.
183
184      C      CDAMP=DAMPING COEFFICIENT
185      C      NDAMP=DAMPING FACTOR
186
187      C      DAMP=SUBROUTINE FOR CALCULATING DEGREE OF DAMPING AND
188      C      CORRESPONDING DAMPED ANGULAR FREQUENCY
189      C      PSTAR=CALCULATED VALUE
190
191      C      DISPL=SUBROUTINE FOR CALCULATING SEGMENT LENGTH
192      C      DELTAX=CALCULATED VALUE
193
194      C      SPEED=SUBROUTINE FOR CALCULATING END VELOCITY
195      C      XDOT1=CALCULATED VALUE
196
197      C      X1=SEGMENT END
198
199
200      210 CONTINUE
201      C      MU=(X**2-1.0)
202      C      NC=2.0
203      CALL DAMP(N,NC,PSTAR)
204      CALL DISPL(X0,XD0T0,N,DELTAT,PSTAR,DELTAX)
205      CALL SPEED(X0,XD0T0,N,DELTAT,PSTAR,DELTAX)
206      X1=X0+DELTAX
207      DIFFX1=XPREV
208      IF(DABS(DIFFX1).LT.1.0D-02) GO TO 220
209      XPREV=X1
210      X=X0+DELTAX/2.0
211      GO TO 210
212
220      220 CONTINUE
221
222      C
223      C      TESTING FOR THE CHANGE IN SIGN OF THE DISPLACEMENT. IF IT IS
224      C      DIFFERENT, THE PROGRAM BEGINS TESTING FOR EITHER A PEAK OR TROUGH.
225      C      STARTING AT STEP 250. IF NOT, THE ITERATION CONTINUES.
226
227
228      CALL SIG(X1,SGNX1)

```

```

221      IF(SGNX1.NE.SGNX0) GO TO 260
222      230 CONTINUE
223      C#####
224      C      TP=ARRAY FOR T0
225      C      AMPL=ARRAY FOR X0
226      C      VP=ARRAY FOR XDOT0
227      C
228      C#####
229      C      TP(I)=T0
230      C      AMPL(I)=X0
231      C      VP(I)=XDOT0
232      C#####
233      C      THE INTERIM VALUES ARE WRITTEN INTO OUTPUT FILE VDPPNUM.
234      C
235      C##### WRITE(10,240) T0,X0,XDOT0
236      C      240 FORMAT(1X,3(G20.10,2X))
237      C#####
238      C      RESETTING THE INITIAL CONDITIONS FOR THE NEXT SEGMENT.
239      C
240      C##### X0=X1
241      C      XDOT0=XDOT1
242      C      T0=T1
243      C      GO TO 200
244      C#####
245      C      X0=X1
246      C      XDOT0=XDOT1
247      C      T0=T1
248      C      GO TO 200
249      C#####
250      C      STARTING AT STEP 260, THE PROGRAM TESTS FOR EITHER A PEAK OR
251      C      TROUGH (I. E., ZERO VELOCITY). IF THE VELOCITY IS ZERO AT THE END
252      C      OF THE CURRENT TIME INTERVAL, THE PROGRAM MOVES TO STEP 370 AND
253      C      STARTS THE CALCULATIONS FOR THE NEXT HALF-CYCLE. IF NOT, A
254      C      POSSIBLE CHANGE OF SIGN FOR THE VELOCITY IS TESTED FOR. SHOULD IT
255      C      BE NEGATIVE, THE PROGRAM GOES TO STEP 230 AND CONTINUES. A
256      C      POSITIVE RESULT ALLOWS THE PROGRAM TO ESTIMATE THE VALUE OF THE
257      C      DISPLACEMENT AT THIS PEAK OR TROUGH, XF (SEE STEPS 310 TO 320).
258      C      THIS VALUE IS COMPARED TO A PREVIOUS VALUE (XPREV), AND IF THE
259      C      DIFFERENCE IS VERY SMALL, THE PROGRAM GOES TO STEP 320 AND
260      C      CONTINUES. IF NOT, XF BECOMES XPREV, THE PARAMETERS ARE RESET,
261      C      AND THE ITERATION CONTINUES AT STEP 310.
262      C
263      C#####
264      C      250 CONTINUE
265      C      IF(DABS(XDOT1).GT.1.0D-08) GO TO 260
266      I=I+1
267      C      TP(I)=T1
268      C      AMPL(I)=X1
269      C      VP(I)=XDOT1
270      C      WRITE(10,260)
271      C      260 FORMAT('O')
272      C      WRITE(10,270) T1,X1,XDOT1
273      C      270 FORMAT(1X,3(G20.10,2X))
274      C      WRITE(10,280)
275      C      280 FORMAT('O')
276      C      SGNX0=-SGNX1
277      C      GO TO 370
278      C
279      C      280 CONTINUE
280      CALL DISPL(X0,XDOT0,N,DELTAT,PSTAR,DELX)
281      CALL SPEED(X0,XDOT0,N,DELTAT,PSTAR,XTOT1)
282      CALL SIG(XDOT1,SGMV1)
283      IF(SGMV1.EQ.SGNX1) GO TO 230
284      C      TP(I)=T0
285      C      AMPL(I)=X0
286      C      VP(I)=XDOT0
287      C      WRITE(10,300) T0,X0,XDOT0
288      C      300 FORMAT(1X,3(G20.10,2X))
289      XF=X0+DELX
290      C      T1=T1-DELTAT
291      C#####
292      C      THE ITERATION FOR EITHER THE PEAK OR TROUGH IS DONE FROM STEPS 310
293      C      TO 320.
294      C
295      C      DELT=ESTIMATED VALUE FOR THE TIME INTERVAL TO REACH ZERO VELOCITY
296      C      TIME=SUBROUTINE FOR CALCULATING THE PRECISE VALUE FOR THIS TIME
297      C      INTERVAL
298      C      DTIME=CALCULATED VALUE
299      C
300      C#####
301      C      310 CONTINUE
302      C      DELT=DELTAT
303      CALL TIME(X0,XDOT0,DELT,N,PSTAR,DTIME)
304      CALL DISPL(X0,XDOT0,N,DTIME,PSTAR,DELX)
305      CALL SPEED(X0,XDOT0,N,DTIME,PSTAR,XTOT1)
306      C      X1=X0+DELX
307      C      DELX=X1-XF
308      C      DELX=X1-XF
309      IF(DABS(DELX).LT.1.0D-08) GO TO 320
310      XF=X1
311      N=MUR((XF**2-1.0)/2.0
312      CALL DAMP(N,PSTAR)
313      GO TO 310
314      C      320 CONTINUE
315      C#####
316      C      THE VALUES FOR THE PEAK OR TROUGH ARE WRITTEN INTO OUTPUT FILE
317      C      VDPPNUM.
318      C
319      C#####
320      C      T1=T1+DELT
321      C      CALL SPEED(X0,XDOT0,N,DTIME,PSTAR,XTOT1)
322      C      320 FORMAT('O')
323      C      320 FORMAT(1X,3(G20.10,2X))
324      C      T0(I)=T1
325      C      AMPL(I)=X1
326      C      VP(I)=XDOT1
327      C      WRITE(10,330)
328      C      330 FORMAT('O')
329      C      WRITE(10,340) T1,X1,XDOT1
330      C      340 FORMAT(1X,3(G20.10,2X))

```

```

331      WRITE(10,360)
332      360 FORMAT(1X,3(G20.10,2X))
333
334      C      TESTING FOR THE NUMBER OF HALF-CYCLES.
335
336      C      IF(K.EQ.NCYCLE) GO TO 380
337
338      C      RESETTING THE INITIAL CONDITIONS FOR THE NEXT HALF-CYCLE.
339
340      C      X0=X1
341      C      XDOT0=Xdot1
342      C      XPREV=X0
343      C      X=X0
344      C      SIGNX=-SIGNXO
345      C      K=K+1.0
346
347      C      ONCE THE PEAK OR TROUGH CO-ORDINATES HAVE BEEN FOUND, THE VALUES
348      C      AT THE END OF THE NEXT TIME INCREMENT ARE CALCULATED, SINCE THE
349      C      FORMER DO NOT OCCUR AT THE END OF THE PREVIOUS ONE. THE
350      C      DISPLACEMENT IS FOUND EXACTLY AS WAS DONE FROM STEPS 210 TO 220,
351      C      EXCEPT WITH THE TIME INTERVAL BEING DIFFT.
352
353      C      DIFFT=DELTAT-DTIME
354      C      T0=T1+DIFFT
355      360 CONTINUE
356      C      MU=(X**2-1.0)
357      C      N=C/2.0
358      CALL DAMP(N,PSTAR)
359      CALL DISPL(X0,XDOT0,N,DIFFT,PSTAR,DELTAT)
360      CALL SPEED(X0,XDOT0,N,DIFFT,PSTAR,XDOT1)
361      X1=X0+DELTAX
362      DIFFX=X1-XPREV
363      IF(DABS(DIFFX).LT.1.0D-08) GO TO 370
364      XPREV=X1
365      X=X0+DELTAX/2.0
366      GO TO 360
367      370 CONTINUE
368      X0=X1
369      XDOT0=Xdot1
370      GO TO 200
371      380 CONTINUE
372      NPOINT=1
373
374      C      THE FOLLOWING SECTION (ENDING AT LINE 380) IS THE RUNGE-KUTTA
375      C      SOLUTION OF THE SAME ORIGINAL EQUATION.
376
377      C      TFINAL=FINAL TIME
378      C      INTVL=NUMBER OF INCREMENTS
379
380      C      TFINAL=T1+Q1
381      C      INTVL=TFINAL/DT+1
382
383      C      THE FOLLOWING ARE PARAMETERS FOR THE RUNGE-KUTTA SUBROUTINE.
384      C      CONSULT THE IMSL MANUAL FOR "OVERK" FOR FURTHER DETAILS.
385
386      C      XRKTIME0
387      C      Y(1)=DISPL0
388      C      Y(2)=VO
389
390      C      SIGHK=LARGEST VALUE OF RUNGE-KUTTA VELOCITY (FOR PLOT SCALING)
391
392      C      NEON=2
393      C      NW=2
394      C      TOL=1.0D-04
395      C      IND1
396      C      XEND=DT
397
398      C      ZEIT=CUMULATIVE VALUE OF TIME AT END OF EACH INCREMENT
399
400      C      ZEIT=0.0
401
402      C      AMPLRK=ARRAY FOR AMPLITUDE VALUES
403      C      TRKARRAY FOR TIME VALUES
404      C      VRKARRAY FOR VELOCITY VALUES
405
406      C      DO 380 J=1,INTVL
407      C      AMPLRK(J)=Y(1)
408      C      TRK(J)=ZEIT
409      C      VRK(J)=Y(2)
410      C      CALL OVERK(NEON,FCH,XRK,Y,XEND,TOL,IND,CH,NW,W,IER)
411
412
413      C      INCREMENTING THE TIME VALUES.
414
415      C      ZEIT=XEND
416      C      XEND=XEND+DT
417
418      380 CONTINUE
419
420      C      THE CALCULATED VALUES ARE WRITTEN INTO OUTPUT FILE VOPNNUM.
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440

```

POOR COPY
COPIE DE QUALITEE INFERIEURE

```

441
442      WRITE(1,500)
443      500 FORMAT('1', 'RUNGE-KUTTA RESULTS FOR THE EQUATION:')
444      WRITE(1,410)
445      410 FORMAT('0', '(DOUBLE-DOT)+MU=(X**2-1)*X(DOT)+X**0')
446      WRITE(1,420) MU
447      420 FORMAT('0', 'MU', .620, 10)
448      WRITE(1,430) DISPL0, V0
449      430 FORMAT('0', '(X0)**', .620, 10, 2X, '(X(DOT))0)**', .620, 10F
450      WRITE(1,440)
451      440 FORMAT(' ', 'X', 'TIME', '10X', 'X(T)', 'X', 'VELOCITY')
452      DO 450 M=1, INTVL
453      WRITE(1,450) TRK(M), AMPLRK(M), VRK(M)
454      450 FORMAT(1X, 3(.620, 10, 2X))
455      450 CONTINUE
456
457      C
458      C   THE FOLLOWING FINDS THE LARGEST MAGNITUDES OF THE DISPLACEMENTS
459      C   AND VELOCITIES FOR THE SOLUTIONS FOR PLOT SCALING PURPOSES.
460
461      C   BIGDP=LARGEST VALUE OF PIECEWISE LINEAR DISPLACEMENT
462      C   BIGVP=LARGEST VALUE OF PIECEWISE LINEAR VELOCITY
463
464      C   BIGDRK=LARGEST VALUE OF RUNGE-KUTTA DISPLACEMENT
465      C   BIGVRK=LARGEST VALUE OF RUNGE-KUTTA VELOCITY
466
467      C
468      BIGDP=DISPL0
469      BIGVP=V0
470      DO 470 IB=1, NPOINT
471      DISP=AMPLP(IB)
472      VEL=VP(IB)
473      IF(DABS(DISP).GE.BIGDP) BIGDP=DABS(DISP)
474      IF(DABS(VEL).GE.BIGVP) BIGVP=DABS(VEL)
475      470 CONTINUE
476      BIGDRK=DISPL0
477      BIGVRK=V0
478      DO 480 IB=1, INTVL
479      DISP=AMPLRK(IB)
480      VEL=VRK(IB)
481      IF(DABS(DISP).GE.BIGDRK) BIGDRK=DABS(DISP)
482      IF(DABS(VEL).GE.BIGVRK) BIGVRK=DABS(VEL)
483      480 CONTINUE
484
485      C
486      C   THE NEXT FEW LINES ARE USED IN FINALIZING ALL THE PLOT PARAMETERS
487      C   WHICH INCLUDES AXIS SCALING.
488
489      C
490      BIGD=BIGDRK
491      IF(BIGD.GE.BIGDRK) BIGD=BIGDP
492      M=1
493      IF(BIGD.LE.1.0) GO TO 500
494      490 CONTINUE
495      DECD=BIGD/(10.0**M)
496      IF(DECD.LE.1.0) GO TO 520
497      M=M+1
498      GO TO 490
499      500 CONTINUE
500      BIGD=BIGD/(10.0**M)
501      IF(BIGD.GE.1.0) GO TO 510
502      M=M+1
503      GO TO 500
504      510 CONTINUE
505      DECD=BIGD/10.0
506      M=(M-1)
507      520 CONTINUE
508      IF(DECD.LE.0.1) VFACT=0.1
509      IF((DECD.GT.0.1).AND.(DECD.LE.0.25)) VFACT=0.25
510      IF((DECD.GT.0.25).AND.(DECD.LE.0.5)) VFACT=0.5
511      IF((DECD.GT.0.5).AND.(DECD.LE.1.0)) VFACT=1.0
512      SFFACT=YCD/10.0
513      YAD=-YFACT+SFFACT*10.0**M
514      DEN=YCD/(2.0*SFFACT)
515      YBD=(YFACT/DEN)=10.0**M
516
517      M=1
518      IF(TFINAL.LE.1.0) GO TO 540
519      530 CONTINUE
520      DECT=TFINAL/(10.0**M)
521      IF(DECT.LE.1.0) GO TO 560
522      M=M+1
523      GO TO 530
524      540 CONTINUE
525      DIGT=TFINAL-(10.0**M)
526      IF(DIGT.GE.1.0) GO TO 550
527      M=M+1
528      GO TO 540
529      560 CONTINUE
530      DECT=DIGT/10.0
531      M=(M-1)
532      580 CONTINUE
533      IF(DECT.LE.0.07) HFACT=0.1
534      IF((DECT.GT.0.07).AND.(DECT.LE.0.14)) HFACT=0.2
535      IF((DECT.GT.0.14).AND.(DECT.LE.0.35)) HFACT=0.5
536      IF((DECT.GT.0.35).AND.(DECT.LE.0.7)) HFACT=1.0
537      IF((DECT.GT.0.7).AND.(DECT.LE.1.0)) HFACT=2.0
538      HBD=HFACT*10.0**M
539      LONG=HBD
540      XAD(2)*DFLOAT(LONG)=HBD-0.005
541
542      BIGV=BIGVRK
543      IF(BIGV.GE.BIGVRK) BIGV=BIGVP
544      M=1
545      IF(BIGV.LE.1.0) GO TO 580
546      570 CONTINUE
547      DCDV=BIGV/(10.0**M)
548      IF(DCDV.LE.1.0) GO TO 600
549      M=M+1
550      GO TO 570

```

```

651   680 CONTINUE
652   DIGV=SIGV*(10.0**M)
653   IF(DIGV.GE.1.0) GO TO 680
654   M=M+1
655   GO TO 680
656   680 CONTINUE
657   DECV=DIGV/10.0
658   M=(M-1)
659   680 CONTINUE
660   IF(DECV.LE.0.1) VFACT=0.1
661   IF((DECV.GT.0.1).AND.(DECV.LE.0.25)) VFACT=0.25
662   IF((DECV.GT.0.25).AND.(DECV.LE.0.5)) VFACT=0.5
663   IF((DECV.GT.0.5).AND.(DECV.LE.1.0)) VFACT=1.0
664   HAP=YBD*(YCD/2.0)
665   HBP=YBD
666   HCP=YCD
667   SFACT=VCP/10.0
668   VAP=-VFACT*SFACT=10.0**M
669   DEN=VCP/(2.0*SFACT)
670   VBP=(VFACT/DEN)=10.0**M
671   XAPH(1)=HAP+0.005
672   XAPH(2)=HAP-0.005
673   YAPH(1)=0.0
674   YAPH(2)=0.0
675   XAPV(1)=0.0
676   XAPV(2)=0.0
677   YAPV(1)=VAP+0.005
678   YAPV(2)=VAP-0.005
679
680 C THE FOLLOWING SECTION SETS UP THE PIECEWISE LINEAR RESULTS SO THAT
681 C BETWEEN 80 AND 80 POINTS ARE PLOTTED, DEPENDING ON HOW CLOSE THE
682 C TOTAL NUMBER OF CALCULATIONS (NPOINT) IS TO AN INTEGRAL MULTIPLE
683 C OF 80.
684 C
685 C YPARRAY FOR PLOTTED PIECEWISE LINEAR DISPLACEMENTS
686 C TPPARRAY FOR PLOTTED PIECEWISE LINEAR TIME VALUES
687 C VPARRAY FOR PLOTTED PIECEWISE LINEAR VELOCITIES
688 C
689 C#####
690 INCPP=NPOINT/80
691 IDRC=DFLOAT(NPOINT/80)-INCP
692 IF(IDEC.GE.0.5) INCP=NPOINT/80+1
693 NPP=1
694 NDP=1
695 810 CONTINUE
696 YP(NPP)=AMPLP(NDP)
697 TPP(NPP)=TP(NDP)
698 VPP(NPP)=VP(NDP)
699 NPP=NPP+1
700 NDP=NDP+INCP
701 IF(NDP.GE.NPOINT) GO TO 820
702 GO TO 810
703 820 CONTINUE
704 IPOINT=(NPP-NPOINT)-INCP/2
705 IHALF=INCP/2
706 IF(IPONT.LE.IHALF) GO TO 830
707 YP(NPP)=AMPLP(NDP)
708 TPP(NPP)=TP(NDP)
709 VPP(NPP)=VP(NDP)
710 GO TO 840
711 830 CONTINUE
712 NPP=NPP-1
713 NDP=NDP-INCP
714 840 CONTINUE
715 C#####
716 C DUE TO LIMITATIONS IN THE PLOT ROUTINES USED FURTHER ON, THE
717 C FOLLOWING SETS UP THE RUNGE-KUTTA RESULTS SO THAT EVERY SECOND
718 C POINT IS PLOTTED IF MORE THAN 1000 POINTS WERE CALCULATED.
719 C
720 C#####
721 INCRK=2
722 IF(INTVL.LE.888) INCRK=1
723 NPK=1
724 NDRK=1
725 850 CONTINUE
726 YRK(NPK)=AMPLRK(NDRK)
727 TPRK(NPK)=TRK(NDRK)
728 VPRK(NPK)=VRK(NDRK)
729 NPK=NPK+1
730 NDRK=NDRK+INCRK
731 IF(NDRK.GE.INTVL) GO TO 880
732 GO TO 850
733 860 CONTINUE
734 IF(INCRK.EQ.1) GO TO 870
735 COUNT=DFLOAT(INTVL)
736 INC=COUNT/2.0
737 DELINC=COUNT/2.0-DFLOAT(INC)
738 IF(DELINC.EQ.0.0) GO TO 880
739 870 CONTINUE
740 YRK(NPK)=AMPLRK(INTVL)
741 TPRK(NPK)=TRK(INTVL)
742 VPRK(NPK)=VRK(INTVL)
743 GO TO 880
744 880 CONTINUE
745 NPK=NPK-1
746 NDRK=NDRK-INCRK
747 880 CONTINUE
748 C#####
749 C PLOTTING THE SOLUTIONS. CONSULT THE WRITEUP ON CGPL/CGPL2 AND
750 C THE MANUAL ON DIGITAL PLOTTING FOR DETAILS.
751 C
752 C#####
753 NDNPBK
754 NF=1
755 CALL CGPL2(TPRK,YRK,ND,NF,S,HAD,HBD,HCD,VAD,VBD,VCD,ALPHD)
756 NDNPBP
757 CALL CGPL2(VPRK,YP,ND,2,1,HAD,HBD,HCD,VAD,VBD,VCD,ALPHD)
758

```

```

681      NF=4
682      CALL CGPL2(XAD,YAD,2,NF,4,HAD,HBD,HCD,VAD,VBD,VCD,ALPHD)
683      HA=HAD
684      HC=HCD
685      VA=VAD
686      VC=VCD
687      IPLOT=0
688 700  CONTINUE
689      STARTX=(HC-8.3)/2.0
690      STARTY=VC+0.5
691      HORIZ=HA
692      VERT=VC+0.5
693      CALL PLOT(HORIZ,VERT,3)
694      HORIZ=HC
695      CALL PLOT(HORIZ,VERT,2)
696      VERT=VERT+4.2
697      CALL PLOT(HORIZ,VERT,2)
698      HORIZ=HA
699      CALL PLOT(HORIZ,VERT,2)
700      VERT=VC+0.5
701      CALL PLOT(HORIZ,VERT,2)
702      HORIZ=STARTX+2.0
703      VERT=STARTY+3.8
704      CALL SYMBOL(HORIZ,VERT,0.2,34,0.0,-1)
705      HORIZ=STARTX+2.4
706      CALL SYMBOL(HORIZ,VERT,0.2,'*',0.0,1)
707      HORIZ=STARTX+2.8
708      CALL NUMBER(HORIZ,VERT,0.2,MU,0.0,4)
709      HORIZ=STARTX+3.2
710      VERT=STARTY+3.1
711      CALL SYMBOL(HORIZ,VERT,0.2,52,0.0,-1)
712      HORIZ=STARTX+4.0
713      VERT=STARTY+2.8
714      CALL SYMBOL(HORIZ,VERT,0.2,'T (PIECEWISE ',0.0,12)
715      HORIZ=STARTX+1.0
716      VERT=STARTY+2.8
717      CALL SYMBOL(HORIZ,VERT,0.2,'LINEARIZATION') '*',0.0,16)
718      HORIZ=STARTX+4.2
719      CALL NUMBER(HORIZ,VERT,0.2,DELTAT,0.0,4)
720      HORIZ=STARTX+0.2
721      VERT=STARTY+2.4
722      CALL SYMBOL(HORIZ,VERT,0.2,52,0.0,-1)
723      HORIZ=STARTX+4.0
724      VERT=STARTY+2.8
725      CALL SYMBOL(HORIZ,VERT,0.2,'RUNGE-KUTTA') '*',0.0,17)
726      HORIZ=STARTX+4.0
727      VERT=STARTY+0.7
728      CALL PLOT(HORIZ,VERT,3)
729      HORIZ=STARTX+0.8
730      VERT=STARTY+1.8
731      CALL SYMBOL(HORIZ,VERT,0.2,'X(0) =',0.0,8)
732      HORIZ=STARTX+1.8
733      CALL NUMBER(HORIZ,VERT,0.2,DISPL0,0.0,4)
734      HORIZ=STARTX+3.8
735      CALL SYMBOL(HORIZ,VERT,0.2,'X(0) =',0.0,8)
736      HORIZ=STARTX+3.825
737      VERT=STARTY+1.8
738      CALL SYMBOL(HORIZ,VERT,0.2,75,0.0,-1)
739      HORIZ=STARTX+4.0
740      VERT=STARTY+1.8
741      CALL NUMBER(HORIZ,VERT,0.2,V0,0.0,4)
742      HORIZ=STARTX+0.3
743      VERT=STARTY+0.7
744      CALL PLOT(HORIZ,VERT,3)
745      HORIZ=STARTX+0.8
746      VERT=STARTY+0.8
747      CALL SYMBOL(HORIZ,VERT,0.2,'RUNGE-KUTTA',0.0,11)
748      HORIZ=STARTX+0.8
749      VERT=STARTY+0.3
750      CALL SYMBOL(HORIZ,VERT,0.2,1,0.0,-1)
751      HORIZ=STARTX+1.0
752      VERT=STARTY+0.2
753      CALL SYMBOL(HORIZ,VERT,0.2,'PIECEWISE LINEARIZATION',0.0,23)
754      IF(IIPLOT.NE.0) GO TO 710
755      NF=0
756      CALL CGPL2(XAD,YAD,2,NF,4,HAD,HBD,HCD,VAD,VBD,VCD,ALPHD)
757      HDP=HPRK
758      HF11
759      CALL CGPL2(YRK,YPRK,ND,NF,5,HAP,HBP,HCP,VAP,VBP,VCP,ALPHP)
760      HDP=HPP
761      CALL CGPL2(YP,VPP,ND,2,1,HAP,HBP,HCP,VAP,VBP,VCP,ALPHP)
762      HF14
763      CALL CGPL2(YAPH,YAPH,2,NF,4,HAP,HBP,HCP,VAP,VBP,VCP,ALPHP)
764      HF14
765      CALL CGPL2(YAPV,YAPV,2,NF,4,HAP,HBP,HCP,VAP,VBP,VCP,ALPHP)
766      HAP=0.0
767      HC=HCP
768      VA=VAP
769      VC=VCP
770      IPLOT=1
771      GO TO 700
772 710  CONTINUE
773      NF=0
774      CALL CGPL2(XAPV,YAPV,2,NF,4,HAP,HBP,HCP,VAP,VBP,VCP,ALPHP)
775      STOP
776      END
777
778      C
779      C
780      C
781      C
782      C
783      C
784      C
785      C
786      C
787      C
788      C
789      C
790      C
791      C
792      C
793      C
794      C
795      C
796      C
797      C
798      C
799      C
800      C
801      C
802      C
803      C
804      C
805      C
806      C
807      C
808      C
809      C
810      C
811      C
812      C
813      C
814      C
815      C
816      C
817      C
818      C
819      C
820      C
821      C
822      C
823      C
824      C
825      C
826      C
827      C
828      C
829      C
830      C
831      C
832      C
833      C
834      C
835      C
836      C
837      C
838      C
839      C
840      C
841      C
842      C
843      C
844      C
845      C
846      C
847      C
848      C
849      C
850      C
851      C
852      C
853      C
854      C
855      C
856      C
857      C
858      C
859      C
860      C
861      C
862      C
863      C
864      C
865      C
866      C
867      C
868      C
869      C
870      C
871      C
872      C
873      C
874      C
875      C
876      C
877      C
878      C
879      C
880      C
881      C
882      C
883      C
884      C
885      C
886      C
887      C
888      C
889      C
890      C
891      C
892      C
893      C
894      C
895      C
896      C
897      C
898      C
899      C
900      C
901      C
902      C
903      C
904      C
905      C
906      C
907      C
908      C
909      C
910      C
911      C
912      C
913      C
914      C
915      C
916      C
917      C
918      C
919      C
920      C
921      C
922      C
923      C
924      C
925      C
926      C
927      C
928      C
929      C
930      C
931      C
932      C
933      C
934      C
935      C
936      C
937      C
938      C
939      C
940      C
941      C
942      C
943      C
944      C
945      C
946      C
947      C
948      C
949      C
950      C
951      C
952      C
953      C
954      C
955      C
956      C
957      C
958      C
959      C
960      C
961      C
962      C
963      C
964      C
965      C
966      C
967      C
968      C
969      C
970      C
971      C
972      C
973      C
974      C
975      C
976      C
977      C
978      C
979      C
980      C
981      C
982      C
983      C
984      C
985      C
986      C
987      C
988      C
989      C
990      C
991      C
992      C
993      C
994      C
995      C
996      C
997      C
998      C
999      C
1000     C
1001     C
1002     C
1003     C
1004     C
1005     C
1006     C
1007     C
1008     C
1009     C
1010     C
1011     C
1012     C
1013     C
1014     C
1015     C
1016     C
1017     C
1018     C
1019     C
1020     C
1021     C
1022     C
1023     C
1024     C
1025     C
1026     C
1027     C
1028     C
1029     C
1030     C
1031     C
1032     C
1033     C
1034     C
1035     C
1036     C
1037     C
1038     C
1039     C
1040     C
1041     C
1042     C
1043     C
1044     C
1045     C
1046     C
1047     C
1048     C
1049     C
1050     C
1051     C
1052     C
1053     C
1054     C
1055     C
1056     C
1057     C
1058     C
1059     C
1060     C
1061     C
1062     C
1063     C
1064     C
1065     C
1066     C
1067     C
1068     C
1069     C
1070     C
1071     C
1072     C
1073     C
1074     C
1075     C
1076     C
1077     C
1078     C
1079     C
1080     C
1081     C
1082     C
1083     C
1084     C
1085     C
1086     C
1087     C
1088     C
1089     C
1090     C
1091     C
1092     C
1093     C
1094     C
1095     C
1096     C
1097     C
1098     C
1099     C
1100     C
1101     C
1102     C
1103     C
1104     C
1105     C
1106     C
1107     C
1108     C
1109     C
1110     C
1111     C
1112     C
1113     C
1114     C
1115     C
1116     C
1117     C
1118     C
1119     C
1120     C
1121     C
1122     C
1123     C
1124     C
1125     C
1126     C
1127     C
1128     C
1129     C
1130     C
1131     C
1132     C
1133     C
1134     C
1135     C
1136     C
1137     C
1138     C
1139     C
1140     C
1141     C
1142     C
1143     C
1144     C
1145     C
1146     C
1147     C
1148     C
1149     C
1150     C
1151     C
1152     C
1153     C
1154     C
1155     C
1156     C
1157     C
1158     C
1159     C
1160     C
1161     C
1162     C
1163     C
1164     C
1165     C
1166     C
1167     C
1168     C
1169     C
1170     C
1171     C
1172     C
1173     C
1174     C
1175     C
1176     C
1177     C
1178     C
1179     C
1180     C
1181     C
1182     C
1183     C
1184     C
1185     C
1186     C
1187     C
1188     C
1189     C
1190     C
1191     C
1192     C
1193     C
1194     C
1195     C
1196     C
1197     C
1198     C
1199     C
1200     C
1201     C
1202     C
1203     C
1204     C
1205     C
1206     C
1207     C
1208     C
1209     C
1210     C
1211     C
1212     C
1213     C
1214     C
1215     C
1216     C
1217     C
1218     C
1219     C
1220     C
1221     C
1222     C
1223     C
1224     C
1225     C
1226     C
1227     C
1228     C
1229     C
1230     C
1231     C
1232     C
1233     C
1234     C
1235     C
1236     C
1237     C
1238     C
1239     C
1240     C
1241     C
1242     C
1243     C
1244     C
1245     C
1246     C
1247     C
1248     C
1249     C
1250     C
1251     C
1252     C
1253     C
1254     C
1255     C
1256     C
1257     C
1258     C
1259     C
1260     C
1261     C
1262     C
1263     C
1264     C
1265     C
1266     C
1267     C
1268     C
1269     C
1270     C
1271     C
1272     C
1273     C
1274     C
1275     C
1276     C
1277     C
1278     C
1279     C
1280     C
1281     C
1282     C
1283     C
1284     C
1285     C
1286     C
1287     C
1288     C
1289     C
1290     C
1291     C
1292     C
1293     C
1294     C
1295     C
1296     C
1297     C
1298     C
1299     C
1300     C
1301     C
1302     C
1303     C
1304     C
1305     C
1306     C
1307     C
1308     C
1309     C
1310     C
1311     C
1312     C
1313     C
1314     C
1315     C
1316     C
1317     C
1318     C
1319     C
1320     C
1321     C
1322     C
1323     C
1324     C
1325     C
1326     C
1327     C
1328     C
1329     C
1330     C
1331     C
1332     C
1333     C
1334     C
1335     C
1336     C
1337     C
1338     C
1339     C
1340     C
1341     C
1342     C
1343     C
1344     C
1345     C
1346     C
1347     C
1348     C
1349     C
1350     C
1351     C
1352     C
1353     C
1354     C
1355     C
1356     C
1357     C
1358     C
1359     C
1360     C
1361     C
1362     C
1363     C
1364     C
1365     C
1366     C
1367     C
1368     C
1369     C
1370     C
1371     C
1372     C
1373     C
1374     C
1375     C
1376     C
1377     C
1378     C
1379     C
1380     C
1381     C
1382     C
1383     C
1384     C
1385     C
1386     C
1387     C
1388     C
1389     C
1390     C
1391     C
1392     C
1393     C
1394     C
1395     C
1396     C
1397     C
1398     C
1399     C
1400     C
1401     C
1402     C
1403     C
1404     C
1405     C
1406     C
1407     C
1408     C
1409     C
1410     C
1411     C
1412     C
1413     C
1414     C
1415     C
1416     C
1417     C
1418     C
1419     C
1420     C
1421     C
1422     C
1423     C
1424     C
1425     C
1426     C
1427     C
1428     C
1429     C
1430     C
1431     C
1432     C
1433     C
1434     C
1435     C
1436     C
1437     C
1438     C
1439     C
1440     C
1441     C
1442     C
1443     C
1444     C
1445     C
1446     C
1447     C
1448     C
1449     C
1450     C
1451     C
1452     C
1453     C
1454     C
1455     C
1456     C
1457     C
1458     C
1459     C
1460     C
1461     C
1462     C
1463     C
1464     C
1465     C
1466     C
1467     C
1468     C
1469     C
1470     C
1471     C
1472     C
1473     C
1474     C
1475     C
1476     C
1477     C
1478     C
1479     C
1480     C
1481     C
1482     C
1483     C
1484     C
1485     C
1486     C
1487     C
1488     C
1489     C
1490     C
1491     C
1492     C
1493     C
1494     C
1495     C
1496     C
1497     C
1498     C
1499     C
1500     C
1501     C
1502     C
1503     C
1504     C
1505     C
1506     C
1507     C
1508     C
1509     C
1510     C
1511     C
1512     C
1513     C
1514     C
1515     C
1516     C
1517     C
1518     C
1519     C
1520     C
1521     C
1522     C
1523     C
1524     C
1525     C
1526     C
1527     C
1528     C
1529     C
1530     C
1531     C
1532     C
1533     C
1534     C
1535     C
1536     C
1537     C
1538     C
1539     C
1540     C
1541     C
1542     C
1543     C
1544     C
1545     C
1546     C
1547     C
1548     C
1549     C
1550     C
1551     C
1552     C
1553     C
1554     C
1555     C
1556     C
1557     C
1558     C
1559     C
1560     C
1561     C
1562     C
1563     C
1564     C
1565     C
1566     C
1567     C
1568     C
1569     C
1570     C
1571     C
1572     C
1573     C
1574     C
1575     C
1576     C
1577     C
1578     C
1579     C
1580     C
1581     C
1582     C
1583     C
1584     C
1585     C
1586     C
1587     C
1588     C
1589     C
1590     C
1591     C
1592     C
1593     C
1594     C
1595     C
1596     C
1597     C
1598     C
1599     C
1600     C
1601     C
1602     C
1603     C
1604     C
1605     C
1606     C
1607     C
1608     C
1609     C
1610     C
1611     C
1612     C
1613     C
1614     C
1615     C
1616     C
1617     C
1618     C
1619     C
1620     C
1621     C
1622     C
1623     C
1624     C
1625     C
1626     C
1627     C
1628     C
1629     C
1630     C
1631     C
1632     C
1633     C
1634     C
1635     C
1636     C
1637     C
1638     C
1639     C
1640     C
1641     C
1642     C
1643     C
1644     C
1645     C
1646     C
1647     C
1648     C
1649     C
1650     C
1651     C
1652     C
1653     C
1654     C
1655     C
1656     C
1657     C
1658     C
1659     C
1660     C
1661     C
1662     C
1663     C
1664     C
1665     C
1666     C
1667     C
1668     C
1669     C
1670     C
1671     C
1672     C
1673     C
1674     C
1675     C
1676     C
1677     C
1678     C
1679     C
1680     C
1681     C
1682     C
1683     C
1684     C
1685     C
1686     C
1687     C
1688     C
1689     C
1690     C
1691     C
1692     C
1693     C
1694     C
1695     C
1696     C
1697     C
1698     C
1699     C
1700     C
1701     C
1702     C
1703     C
1704     C
1705     C
1706     C
1707     C
1708     C
1709     C
1710     C
1711     C
1712     C
1713     C
1714     C
1715     C
1716     C
1717     C
1718     C
1719     C
1720     C
1721     C
1722     C
1723     C
1724     C
1725     C
1726     C
1727     C
1728     C
1729     C
1730     C
1731     C
1732     C
1733     C
1734     C
1735     C
1736     C
1737     C
1738     C
1739     C
1740     C
1741     C
1742     C
1743     C
1744     C
1745     C
1746     C
1747     C
1748     C
1749     C
1750     C
1751     C
1752     C
1753     C
1754     C
1755     C
1756     C
1757     C
1758     C
1759     C
1760     C
1761     C
1762     C
1763     C
1764     C
1765     C
1766     C
1767     C
1768     C
1769     C
1770     C
1771     C
1772     C
1773     C
1774     C
1775     C
1776     C
1777     C
1778     C
1779     C
1780     C
1781     C
1782     C
1783     C
1784     C
1785     C
1786     C
1787     C
1788     C
1789     C
1790     C
1791     C
1792     C
1793     C
1794     C
1795     C
1796     C
1797     C
1798     C
1799     C
1800     C
1801     C
1802     C
1803     C
1804     C
1805     C
1806     C
1807     C
1808     C
1809     C
1810     C
1811     C
1812     C
1813     C
1814     C
1815     C
1816     C
1817     C
1818     C
1819     C
1820     C
1821     C
1822     C
1823     C
1824     C
1825     C
1826     C
1827     C
1828     C
1829     C
1830     C
1831     C
1832     C
1833     C
1834     C
1835     C
1836     C
1837     C
1838     C
1839     C
1840     C
1841     C
1842     C
1843     C
1844     C
1845     C
1846     C
1847     C
1848     C
1849     C
1850     C
1851     C
1852     C
1853     C
1854     C
1855     C
1856     C
1857     C
1858     C
1859     C
1860     C
1861     C
1862     C
1863     C
1864     C
1865     C
1866     C
1867     C
1868     C
1869     C
1870     C
1871     C
1872     C
1873     C
1874     C
1875     C
1876     C
1877     C
1878     C
1879     C
1880     C
1881     C
1882     C
1883     C
1884     C
1885     C
1886     C
1887     C
1888     C
1889     C
1890     C
1891     C
1892     C
1893     C
1894     C
1895     C
1896     C
1897     C
1898     C
1899     C
1900     C
1901     C
1902     C
1903     C
1904     C
1905     C
1906     C
1907     C
1908     C
1909     C
1910     C
1911     C
1912     C
1913     C
1914     C
1915     C
1916     C
1917     C
1918     C
1919     C
1920     C
1921     C
1922     C
1923     C
1924     C
1925     C
1926     C
1927     C
1928     C
1929     C
1930     C
1931     C
1932     C
1933     C
1934     C
1935     C
1936     C
1937     C
1938     C
1939     C
1940     C
1941     C
1942     C
1943     C
1944     C
1945     C
1946     C
1947     C
1948     C
1949     C
1950     C
1951     C
1952     C
1953     C
1954     C
1955     C
1956     C
1957     C
1958     C
1959     C
1960     C
1961     C
1962     C
1963     C
1964     C
1965     C
1966     C
1967     C
1968     C
1969     C
1970     C
1971     C
1972     C
1973     C
1974     C
1975     C
1976     C
1977     C
1978     C
1979     C
1980     C
1981     C
1982     C
1983     C
1984     C
1985     C
1986     C
1987     C
1988     C
1989     C
1990     C
1991     C
1992     C
1993     C
1994     C
1995     C
1996     C
1997     C
1998     C
1999     C
2000     C
2001     C
2002     C
2003     C
2004     C
2005     C
2006     C
2007     C
2008     C
2009     C
2010     C
2011     C
2012     C
2013     C
2014     C
2015     C
2016     C
2017     C
2018     C
2019     C
2020     C
2021     C
2022     C
2023     C
2024     C
2025     C
2026     C
2027     C
2028     C
2029     C
2030     C
2031     C
2032     C
2033     C
2034     C
2035     C
2036     C
2037     C
2038     C
2039     C
2040     C
2041     C
2042     C
2043     C
2044     C
2045     C
2046     C
2047     C
2048     C
2049     C
2050     C
2051     C
2052     C
2053     C
2054     C
2055     C
2056     C
2057     C
2058     C
2059     C
2060     C
2061     C
2062     C
2063     C
2064     C
2065     C
2066     C
2067     C
2068     C
2069     C
2070     C
2071     C
2072     C
2073     C
2074     C
2075     C
2076     C
2077     C
2078     C
2079     C
2080     C
2081     C
2082     C
2083     C
2084     C
2085     C
2086     C
2087     C
2088     C
2089     C
2090     C
2091     C
2092     C
2093     C
2094     C
2095     C
2096     C
2097     C
2098     C
2099     C
2100     C
2101     C
2102     C
2103     C
2104     C
2105     C
2106     C
2107     C
2108     C
2109     C
2110     C
2111     C
2112     C
2113     C
2114     C
2115     C
2116     C
2117     C
2118     C
2119     C
2120     C
2121     C
2122     C
2123     C
2124     C
2125     C
2126     C
2127     C
2128     C
2129     C
2130     C
2131     C
2132     C
2133     C
2134     C
2135     C
2136     C
2137     C
2138     C
2139     C
2140     C
2141     C
2142     C
2143     C
2144     C
2145     C
2146     C
2147     C
2148     C
2149     C
2150     C
2151     C
2152     C
2153     C
2154     C
2155     C
2156     C
2157     C
2158     C
2159     C
2160     C
2161     C
2162     C
2163     C
2164     C
2165     C
2166     C
2167     C
2168     C
2169     C
2170     C
2171     C
2172     C
2173     C
2174     C
2175     C
2176     C
2177     C
2178     C
2179     C
2180     C
2181     C
2182     C
2183     C
2184     C
2185     C
2186     C
2187     C
2188     C
2189     C
2190     C
2191     C
2192     C
2193     C
2194     C
2195     C
2196     C
2197     C
2198     C
2199     C
2200    
```

```

771      SUBROUTINE SIG(X,SGN)
772      IMPLICIT REAL*8 (A-H,O-Z)
773      IF(DABS(X).LT.1.0D-08) GO TO 800
774      SGNX=DABS(X)
775      GO TO 810
776 800 CONTINUE
777      SGN0=DO
778 810 CONTINUE
779      RETURN
780      END
781      C
782      C
783      C#####
784      C
785      C      THE FOLLOWING PROGRAM TESTS FOR THE DEGREE OF DAMPING
786      C      (UNDERDAMPING, CRITICAL DAMPING, OR OVERDAMPING) AND THEN
787      C      CALCULATES THE CORRESPONDING DAMPED ANGULAR FREQUENCY.
788      C
789      C#####
790      SUBROUTINE DAMP(N,PSTAR)
791      IMPLICIT REAL*8 (A-H,O-Z)
792      REAL*8 N
793      S0=1.0-N**2
794      CALL SIG(SQ,SGNSQ)
795      IF(SGNSQ.GE.1.0D-08) PSTAR=DSQRT(SQ)
796      IF(DABS(SGNSQ).LT.1.0D-08) PSTAR=1.0
797      IF(SGNSQ.LE.-1.0D-08) PSTAR=DSQRT(-SQ)
798      RETURN
799 800      END
801      C
802      C
803 804      C#####
805      C
806      C      THE FOLLOWING SUBROUTINE CALCULATES THE DISPLACEMENT AT THE END
807      C      OF THE SEGMENT BEING EXAMINED.
808 809      C#####
810      SUBROUTINE DISPL(X0,XDOT0,N,DELTAT,P,DELTAX)
811      IMPLICIT REAL*8 (A-H,O-Z)
812      REAL*8 N
813      C1=X0
814      C2=(N=X0+XDOT0)/P
815      C
816      C      TESTING FOR THE DEGREE OF DAMPING OF THE LINEAR APPROXIMATION. IF
817      C      UNDERDAMPED, THE PROGRAM MOVES TO STEP 800. IF THERE IS CRITICAL
818      C      DAMPING, IT GOES TO STEP 810. FOR OVERDAMPING, THE PROGRAM
819      C      CONTINUES AT STEP 820.
820 821      C
822 823      S0=1.0-N**2
824      IF(SQ.GE.1.0D-08) GO TO 800
825      IF(DABS(SQ).LT.1.0D-08) GO TO 810
826      IF(SQ.LE.-1.0D-08) GO TO 820
827 800 CONTINUE
828      DELTAX=DEXP(-N*DELTAT)*(C1=DCOS(P*DELTAT)+C2=DSIN(P*DELTAT))-X0
829      GO TO 830
830 810 CONTINUE
831      DELTAX=DEXP(-P*DELTAT)*(X0+(N=X0+XDOT0)*DELTAT)-X0
832      GO TO 830;
833 820 CONTINUE
834      DELTAX=DEXP(-N*DELTAT)*(C1=DCOSH(P*DELTAT)+C2=DSINH(P*DELTAT))-X0
835 830 CONTINUE
836      RETURN
837      END
838      C
839      C
840 841      C#####
842      C
843      C      THE FOLLOWING SUBROUTINE CALCULATES THE VELOCITY AT THE END OF THE
844      C      SEGMENT BEING EXAMINED.
845 846      C#####
847      SUBROUTINE SPEED(X0,XDOT0,N,DELTAT,P,XDOT1)
848      IMPLICIT REAL*8 (A-H,O-Z)
849      REAL*8 N
850 851      C
852      C      TESTING FOR THE DEGREE OF DAMPING OF THE LINEAR APPROXIMATION. IF
853      C      UNDERDAMPED, THE PROGRAM MOVES TO STEP 1000. IF THERE IS CRITICAL
854      C      DAMPING, IT GOES TO STEP 1010. FOR OVERDAMPING, THE PROGRAM
855      C      CONTINUES AT STEP 1030.
856 857      C
858      S0=1.0-N**2
859      IF(SQ.GE.1.0D-08) GO TO 1000
860      IF(DABS(SQ).LT.1.0D-08) GO TO 1010
861      IF(SQ.LE.-1.0D-08) GO TO 1020
862 1000 CONTINUE
863      C1=XDOT0
864      C2=-N*(N=X0+XDOT0)/P-P*X0
865      XDOT1=DEXP(-N*DELTAT)*(C1=DCOS(P*DELTAT)+C2=DSIN(P*DELTAT))
866      GO TO 1030
867 1010 CONTINUE
868      XDOT1=DEXP(-P*DELTAT)*(-P*X0+(N=X0+XDOT0)*(1.0-P*DELTAT))
869      GO TO 1030
870 1020 CONTINUE
871      C1=XDOT0
872      C2=-N*(N=X0+XDOT0)/P+P*X0
873      XDOT1=DEXP(-N*DELTAT)*(C1=DCOSH(P*DELTAT)+C2=DSINH(P*DELTAT))
874 1030 CONTINUE
875      RETURN
876      END
877      C
878      C
879      C
880 881      C#####

```

```

881 C THE FOLLOWING SUBROUTINE CALCULATES THE TIME INTERVAL REQUIRED TO
882 C REACH ZERO VELOCITY (I. E., A PEAK OR TROUGH ON THE DISPLACEMENT-
883 C TIME CURVE) BY USING A NEWTON-RAPHSON METHOD.
884 C
885 C#####
886 C SUBROUTINE TIME(X0,XDDOT0,DELT,N,P,DTIME)
887 C IMPLICIT REAL*8 (A-H,D-Z)
888 C REAL*8 N
889 C DELTA=DELT*0.05
890 C
891 C#####
892 C TESTING FOR THE DEGREE OF DAMPING OF THE LINEAR APPROXIMATION. IF
893 C UNDERDAMPED, THE PROGRAM MOVES TO STEP 1100. IF THERE IS CRITICAL
894 C DAMPING, IT GOES TO STEP 1120. FOR OVERTDAMPING, THE PROGRAM
895 C CONTINUES AT STEP 1140.
896 C
897 C#####
898 C011.0-N*P*2
899 IF(SQ.GE.1.0D-08) GO TO 1100
900 IF(DABS(SQ).LT.1.0D-08) GO TO 1120
901 IF(SQ.LE.-1.0D-08) GO TO 1140
902 C#####
903 C PMAIN FUNCTION (VELOCITY)
904 C PP=F-PRIME (ACCELERATION)
905 C
906 C#####
907 1100 CONTINUE
908 C1=XDDOT0
909 C2=N=(N=X0+XDDOT0)/P-P*X0
910 C3=N=C1+P*C2
911 C4=P+C1-N*C2
912 C1110 CONTINUE
913 F=P*EXP(-N*DELT)=(C1+DCOS(P*DELT)+C2*D SIN(P*DELT))
914 FPP=P*EXP(-N*DELT)=(C3+DCOS(P*DELT)+C4*D SIN(P*DELT))
915 DELTA1=DELT-F/FP
916 DIFF=DELT1-DELT
917 IF(DABS(DIFF).LT.1.0D-08) GO TO 1180
918 DELTA=DELT1
919 GO TO 1110
920 C1120 CONTINUE
921 F=P-X0+(N*X0+XDDOT0)=(1.0-F*DELT)
922 FPP=P*(X0-N*X0+XDDOT0)=(2.0-F*DELT)
923 C1130 CONTINUE
924 DELTA1=DELT-F/FP
925 DIFF=DELT1-DELT
926 IF(DABS(DIFF).LT.1.0D-08) GO TO 1180
927 DELTA=DELT1
928 GO TO 1130
929 C1140 CONTINUE
930 C1=XDDOT0
931 C2=N=(N=X0+XDDOT0)/P-P*X0
932 C3=N=C1+P*C2
933 C4=P+C1-N*C2
934 C1150 CONTINUE
935 F=P*EXP(-N*DELT)=(C1+DCOSH(P*DELT)+C2*D SINH(P*DELT))
936 FPP=P*EXP(-N*DELT)=(C3+DCOSH(P*DELT)+C4*D SINH(P*DELT))
937 DELTA1=DELT-F/FP
938 DIFF=DELT1-DELT
939 IF(DABS(DIFF).LT.1.0D-08) GO TO 1180
940 DELTA=DELT1
941 GO TO 1150
942 C1160 CONTINUE
943 DTIME=DELT
944 RETURN
945 END
946 C
947 C
948 C
949 C
950 C
951 C#####
952 C THE FOLLOWING SUBROUTINE CONTAINS THE DIFFERENTIAL EQUATIONS TO BE
953 C SOLVED BY OVERK: THE INSTANTANEOUS VELOCITY AND THE ORIGINAL
954 C EQUATION.
955 C
956 C#####
957 C#####
958 C SUBROUTINE FCN(N,X,Y,YPRIME)
959 C IMPLICIT REAL*8 (A-H,D-Z)
960 C REAL*8 MU
961 C REAL*8 Y(N),YPRIME(N),X
962 C COMMON MU
963 C YPRIME(1)=Y(2)
964 C YPRIME(2)=-(MU*(Y(1))**2-1.0)*YPRIME(1)+Y(1)
965 C RETURN
966 C
967 C#####
END OF FILE

```

POOR COPY
COPIE DE QUALITE INFERIEURE

```

1      C#####
2      C#
3      C# PROGRAM MATHIEU
4      C#
5      C#
6      C#
7      C# THE FOLLOWING PROGRAM CALCULATES THE INTEGRAL ORDER SOLUTION
8      C# FOR THE MATHIEU EQUATION:
9      C# Y(DOUBLE-PRIME)+(A-2*B*COS(2*Z))*Y=0
10     C#
11     C# OBTAINED BY PIECEWISE LINEARIZATION USING CHORDS
12     C#
13     C#
14     C# THE RESULTS ARE BASED ON THE STANDARD FORM:
15     C#
16     C# Y(ALPHA)*COS(P*Z)+BETA*SIN(P*Z)
17     C#
18     C# P=SQRT(A-2*B*COS(2*Z))
19     C#
20     C#
21     C#
22     C# THE CHARACTERISTIC NUMBER, A, IS FOUND BY MAKING AN INITIAL
23     C# GUESS AND FINDING THE CLOSEST VALUE (EITHER ODD OR EVEN
24     C# ORDER) USING A METHOD OF CONTINUED FRACTIONS, DESCRIBED
25     C# FURTHER ON.
26     C#
27     C#
28     C# THE PROGRAM ALSO CALCULATES THE APPROPRIATE MATHIEU FUNCTIONS
29     C# USING A SERIES APPROXIMATION, AS WELL AS A RUNGE-KUTTA
30     C# SOLUTION USING FIFTH- AND SIXTH-ORDER METHODS. ALL THREE
31     C# RESULTS ARE PLOTTED.
32     C#
33     C#####
34     C#
35     C#
36     C#
37     C#
38     C#
39     C#
40     C#
41     C#
42     C#
43     C#
44     C# IMPLICIT REAL*8 (A-H,D-Z)
45     C# EXTERNAL FCK
46     C# REAL*8 Y(Z),C(24),W(Z,Z0)
47     C# REAL*8 LOW,LOWOUT,MID,MIDOUT
48     C# REAL*8 RAT40(1000),RAT02(1000),CDEF(1000),T(1000)
49     C# REAL*8 AMPLF(1000),AMPLS(1000),AMPLRK(1000),Z(1000)
50     C# REAL*8 YP(1000),ZP(1000),YRK(1000),ZPRK(1000)
51     C# REAL*4 HA,HB,HC,VA,VB,YC
52     C# REAL*4 XA(2),YA(2)
53     C# INTEGER*4 ALPH(20)
54     C# COMMON/CALC/AIN,DISPLO,DIFF,ADUT,IPAR
55     C# COMMON/CHAR/A
56     C# COMMON/PARAM/O
57     C#####
58     C#
59     C#
60     C#
61     C#
62     C# THE FOLLOWING VALUES ARE READ FROM DATA FILE MATHIEUDATA:
63     C#
64     C#
65     C# AGUESS=INITIAL GUESS FOR CHARACTERISTIC NUMBER
66     C# (SAME AS IN EQUATION)
67     C#
68     C# DISPLO=INITIAL AMPLITUDE
69     C# V0=INITIAL VELOCITY
70     C# T0=INITIAL TIME
71     C#
72     C# DELTA=INCREMENT/DECREMENT FOR A (USED IN SCAN PHASE)
73     C# H=NUMBER OF DIVISIONS OF PI (AN INTEGRAL MULTIPLE OF 50.0, USED
74     C# FOR FINDING INCREMENT OF PI)
75     C#
76     C# HA,HC=PLOT PARAMETERS FOR HORIZONTAL AXIS (HC AN INTEGRAL MULTIPLE
77     C# OF 10.0)
78     C#
79     C# VC=PLOT PARAMETER FOR VERTICAL AXIS (AN INTEGRAL MULTIPLE OF 12.0,
80     C# 12.0 GIVING BEST RESULTS)
81     C#
82     C# XA,YA=PARAMETERS FOR PLOTTING ZERO LINE
83     C#
84     C# ALPH=ARRAY FOR PLOT AXIS LABELS
85     C#
86     C# READ(5,10) AGUESS,0
87     C# 10 FORMAT(2D10.6)
88     C# READ(5,20) DISPLO,V0,Z0
89     C# 20 FORMAT(3D10.6)
90     C# READ(5,30) DELTA,H
91     C# 30 FORMAT(2D10.4)
92     C# READ(5,40) HA,HC
93     C# 40 FORMAT(2D10.4)
94     C# READ(5,50) VC
95     C# 50 FORMAT(F10.4)
96     C# READ(5,60) XA(1)
97     C# 60 FORMAT(F10.4)
98     C# READ(5,70) YA(1),YA(2)
99     C# 70 FORMAT(2F10.4)
100    C# READ(5,80) (ALPH(I),I=1,12)
101    C# 80 FORMAT(12A4)
102    C# READ(5,90) (ALPH(I),I=13,16)
103    C# 90 FORMAT(4A4)
104    C# READ(5,100) (ALPH(I),I=17,20)
105    C# 100 FORMAT(4A4)
106    C#
107    C# DELTAZ=INCREMENT OF Z (SINCE SOLUTIONS ARE PERIODIC IN EITHER PI
108    C# OR 2*PI)
109    C#
110    C# PI=3.141592653589793

```

```

111      DELTAZ=PI/H
112
113      C THE FOLLOWING ARE USED AS PROGRAM FLAGS FOR CALCULATING EITHER THE
114      C ODD OR EVEN INTEGRAL ORDER CHARACTERISTIC NUMBERS. THE ODD ONE IS
115      C FOUND FIRST AND THEN THE EVEN.
116
117      C IPAR=FLAG FOR DETERMINING ORDER ("0" INDICATES ODD, "1" EVEN)
118      C I=COUNTER FOR DETERMINING HOW MANY TIMES THE SET OF CALCULATIONS
119      C HAVE BEEN DONE ("1" MEANS THE ODD ORDER VALUE IS BEING FOUND,
120      C "2" THE EVEN)
121
122      C
123      C IPAR=0
124      C I=1
125
126      C THE EQUATION PARAMETERS ARE WRITTEN INTO OUTPUT FILE MATHIEUMUM.
127
128      C
129      C WRITE(10,110)
130      110 FORMAT('1','RESULTS FOR THE EQUATION')
131      C WRITE(10,120)
132      120 FORMAT('0',Y'(DOUBLE-PRIME)*(A-2+0.5COS(2*Z))=Y0')
133      C WRITE(10,130) DISPL0,Y0
134      130 FORMAT('0','Y(0)=',1X,G20.10,ZX,'Y(PRIME)(0)=',1X,G20.10)
135      C WRITE(10,140) AGUESS,0
136      140 FORMAT('0','AGUESS)=',1X,G20.10,ZX,'0=',1X,G20.10)
137
138      C
139      C THE FOLLOWING SECTION (ENDING AT STEP 460) CALCULATES THE
140      C CHARACTERISTIC NUMBER FOR A GIVEN AGUESS AND 0, BASED UPON THE
141      C INITIAL DISPLACEMENT AND THE PROXIMITY OF AGUESS TO THE
142      C APPROPRIATE ODD AND EVEN ORDER RESULTS.
143
144      C INITIALLY, A RANGE OF VALUES BETWEEN WHICH THE FINAL RESULT FOR
145      C THE PARTICULAR ORDER IN QUESTION LIES IS FOUND (STEPS 180 TO 300).
146      C THIS IS DONE BY EITHER INCREMENTING OR DECREMENTING AGUESS BY A
147      C FIXED AMOUNT UNTIL THE LIMITS ARE FOUND, WITH THE DIRECTION OF THE
148      C SEARCH (EITHER ABOVE OR BELOW AGUESS) DEPENDING UPON THE OUTPUT
149      C VALUE FROM A SET OF RECURRANCE RELATIONS USING AGUESS AS THE
150      C INPUT.
151
152      C
153      C ONCE THE BOUNDS ARE KNOWN, THE PROGRAM FOCUSES TO THE FINAL RESULT
154      C BY INTERVAL HALVING (STEPS 300 TO 350). HERE THE DIRECTION IS
155      C DETERMINED BY THE OUTPUT VALUE FROM THE SAME RECURRANCE RELATIONS
156      C USING THE FIRST MIDPOINT AS THE INITIAL INPUT.
157
158      C
159      C ONCE THE CHARACTERISTIC NUMBERS FOR THE ODD AND EVEN ORDERS HAVE
160      C BEEN FOUND, THE ONE CLOSEST TO AGUESS IS THE VALUE THAT IS USED
161      C IN CALCULATING THE SOLUTION.
162
163
164
165      C AIN=INPUT VALUE FOR RECURRANCE RELATIONS
166      C RECUR=SUBROUTINE FOR RECURRANCE RELATIONS
167      C DIFF=DIFFERENCE BETWEEN AIN AND OUTPUT VALUE
168
169
170      C
171      C ***NOTE***IF AT ANY POINT THE INPUT AND OUTPUT VALUES ARE NEARLY EQUAL,
172      C THE SEARCH FOR THE UPPER AND LOWER LIMITS OR THE FOCUSING PHASE
173      C STOPS. THE VALUE FOR IPAR IS FOUND (WHICH DETERMINES THE
174      C PERIOD), AND THE OUTPUT VALUE TAKEN AS THE FINAL RESULT.
175
176      C
177      C 180 CONTINUE
178      C AIN=AGUESS
179      C CALL RECUR
180      C IF(DABSH(DIFF),LE,1,OD-08) GO TO 410
181
182      C
183      C THE FOLLOWING CONSIDERS A POSITIVE INPUT VALUE FOR THE
184      C CHARACTERISTIC NUMBER (STEPS 140 TO 230).
185
186      C AOUT=OUTPUT VALUE FROM RECUR
187      C UP=UPPER LIMIT
188      C LOW=LOWER LIMIT
189
190      C
191      C 1. AOUT POSITIVE
192
193      C IF AIN IS GREATER THAN AOUT (SEE STEP 180), AIN IS DECREMENTED AND
194      C A NEW AOUT CALCULATED (SHOULD AIN BECOME LESS THAN ZERO, THE
195      C PROGRAM CONTINUES AT STEP 230). IF AIN IS GREATER THAN AOUT, THE
196      C PROGRAM GOES TO STEP 180. IF AOUT IS POSITIVE, THE PROGRAM
197      C RETURNS TO STEP 180. IF AIN IS LESS THAN AOUT OR IF AIN IS
198      C GREATER THAN AOUT (AOUT BEING NEGATIVE), STEP 200 IS NEXT. THE
199      C LOWER BOUND IS THE CURRENT AIN, THE UPPER LIMIT THE PREVIOUS ONE,
200      C AND THE PROGRAM MOVES TO STEP 300.
201
202      C IF AIN IS LESS THAN AOUT, AIN IS INCREMENTED AND A NEW AOUT FOUND
203      C UNTIL AIN IS GREATER THAN AOUT (STARTING AT STEP 210, RETURNING
204      C TO STEP 210 IF NOT). THE UPPER BOUND IS THE CURRENT AIN AND THE
205      C LOWER THE PREVIOUS ONE. THE PROGRAM THEN MOVES TO STEP 300 TO
206      C BEGIN THE FINAL SET OF ITERATIONS.
207
208
209      C 2. AOUT NEGATIVE
210
211      C AIN IS DECREMENTED (STARTING AT STEP 220) AND A NEW AOUT IS FOUND
212      C UNTIL AOUT BECOMES POSITIVE, THE PROGRAM CONTINUING AT STEP 170.
213      C IF AOUT IS STILL NEGATIVE, IT GOES BACK TO STEP 220. IF AIN GETS
214      C LESS THAN ZERO, THE PROGRAM GOES TO STEP 230.
215
216      C
217      C IF(AIN.LE.0.0) GO TO 230
218      180 CONTINUE
219      C IF(AOUT.LE.0.0) GO TO 220
220      170 CONTINUE

```

```

1 IF(AIN.LE.AOUT) GO TO 210
2 CONTINUE
3 AIN=AIN-DELTA
4 IF(AIN.LE.0.00) GO TO 230
5 CALL RECUR
6 IF(DABS(DIFF).LE.1.00-08) GO TO 410
7 IF(AIN.GT.AOUT) GO TO 180
8 GO TO 200
9 CONTINUE
10 IF(AOUT.GT.0.00) GO TO 180
11 CONTINUE
12 UP=AIN-DELTA
13 LOW=AIN
14 GO TO 300
15 CONTINUE
16 AIN=AIN+DELTA
17 CALL RECUR
18 IF(DABS(DIFF).LE.1.00-08) GO TO 410
19 IF(AIN.LE.AOUT) GO TO 210
20 UP=AIN
21 LOW=AIN-DELTA
22 GO TO 300
23 CONTINUE
24 AIN=AIN-DELTA
25 IF(AIN.LE.0.00) GO TO 230
26 CALL RECUR
27 IF(DABS(DIFF).LE.1.00-08) GO TO 410
28 IF(AIN.LE.AOUT) GO TO 210
29 UP=AIN
30 LOW=AIN-DELTA
31 GO TO 300
32 CONTINUE
33 AIN=AIN-DELTA
34 IF(AIN.LE.0.00) GO TO 230
35 CALL RECUR
36 IF(DABS(DIFF).LE.1.00-08) GO TO 410
37 IF(AOUT.GT.0.00) GO TO 170
38 GO TO 220
39 THE FOLLOWING APPLIES FOR A NEGATIVE AIN (STEPS 230 TO 300).
40
41 1. AOUT NEGATIVE
42
43 IF AIN IS LESS THAN AOUT (STARTING AT STEP 280), AIN IS
44 INCREMENTED AND A NEW AOUT FOUND, THE PROGRAM GOING TO STEP 180
45 FOR A POSITIVE VALUE OF AIN. IF AIN IS LESS THAN AOUT, THE
46 PROGRAM MOVES TO STEP 280. IF AOUT IS STILL NEGATIVE, THE NEXT
47 IS TO STEP 270. THE UPPER LIMIT IS THE CURRENT AIN AND THE
48 LOWER THE PREVIOUS ONE. THE FINAL STAGE THEN STARTS AT STEP
49 300.
50
51 IF AIN IS GREATER THAN AOUT, AIN IS DECREMENTED (STARTING AT STEP
52 280) AND A NEW AOUT FOUND UNTIL AIN IS LESS THAN AOUT, THE PROGRAM
53 RETURNING TO STEP 280 IF NOT. THE UPPER LIMIT IS THE PREVIOUS AIN
54 AND THE LOWER THE CURRENT ONE. STEP 300 IS NEXT.
55
56 2. AOUT POSITIVE
57
58 STARTING AT STEP 280, AIN IS INCREMENTED AND A NEW AOUT IS FOUND
59 UNTIL AOUT IS NEGATIVE, THEN MOVING TO STEP 240 AND CONTINUING
60 FROM THERE. IF AOUT REMAINS POSITIVE, THE PROGRAM RETURNS TO STEP
61 280. SHOULD AIN BECOME POSITIVE, THE PROGRAM MOVES TO STEP 280.
62
63 230 CONTINUE
64 IF(AOUT.GT.0.00) GO TO 280
65 240 CONTINUE
66 IF(AIN.GT.AOUT) GO TO 280
67
68 250 CONTINUE
69 AIN=AIN+DELTA
70 IF(AIN.GT.0.00) GO TO 180
71 CALL RECUR
72 IF(DABS(DIFF).LE.1.00-08) GO TO 410
73 IF(AIN.LE.AOUT) GO TO 260
74 GO TO 270
75 260 CONTINUE
76 IF(AOUT.LE.0.00) GO TO 250
77 270 CONTINUE
78 UP=AIN
79 LOW=AIN-DELTA
80 GO TO 300
81
82 280 CONTINUE
83 AIN=AIN-DELTA
84 CALL RECUR
85 IF(DABS(DIFF).LE.1.00-08) GO TO 410
86 IF(AIN.GT.AOUT) GO TO 280
87 UP=AIN-DELTA
88 LOW=AIN
89 GO TO 300
90
91 290 CONTINUE
92 AIN=AIN+DELTA
93 IF(AIN.GT.0.00) GO TO 180
94 CALL RECUR
95 IF(DABS(DIFF).LE.1.00-08) GO TO 410
96 IF(AOUT.LE.0.00) GO TO 240
97 GO TO 280
98
99 THE FOLLOWING SECTION IS THE FINAL SET OF ITERATIONS TO FIND THE
100 CHARACTERISTIC NUMBER BY USING INTERVAL HALVING.
101
102 MID=INTERVAL.MIDPOINT (AVERAGE OF UP AND LOW)
103 MIDOUT=OUTPUT FROM RECUR USING MID AS INPUT
104
105 BASICALLY, IF MID IS GREATER THAN MIDOUT, MID BECOMES THE NEW
106 UPPER LIMIT AND IF LESS THAN MIDOUT, THE NEW LOWER BOUND.
107
108 300 CONTINUE

```

POOR COPY
COPIE DE QUALITEE INFERIEURE

```

331
332      MID=(U+LOW)/2.0
333      AIM=MID
334      CALL RECUR
335      IF(DABS( DIFF) .LE. 1.0D-08) GO TO 380
336      MIDOUT=MID
337      IF(MID.LE.0.0D) GO TO 330
338      IF(MIDOUT.LE.0.0D) GO TO 320
339      IF(MID.LE.MIDOUT) GO TO 320
340
341      310 CONTINUE
342      UP=MID
343      GO TO 300
344      320 CONTINUE
345      LOW=MID
346      GO TO 300
347      330 CONTINUE
348      IF(MIDOUT.LE.0.0D) GO TO 340
349      GO TO 320
350      340 CONTINUE
351      IF(MID.LE.MIDOUT) GO TO 320
352      GO TO 310
353
354      C
355      C   FROM STEPS 350 TO 380, THE PROGRAM DETERMINES IF BOTH THE ODD AND
356      C   EVEN ORDER SOLUTIONS HAVE BEEN CALCULATED, AND THEN WRITING OUT
357      C   THE APPROPRIATE VALUES INTO OUTPUT FILE MATHIEUMUN once this had
358      C   BEEN DONE.
359
360      C   ADDD+EODD ORDER CHARACTERISTIC NUMBER
361      C   AEVEN+EVEN ORDER NUMBER
362
363      360 CONTINUE
364      IF(I.NE.1) GO TO 380
365      ADDD=AOUT
366      IPAR=1
367      I=2
368      GO TO 380
369      380 CONTINUE
370      AEVEN=AOUT
371      WRITE(10,370) ADDD,AEVEN
372      370 FORMAT(10,'A(ODD)=',1X,G20.10,2X,'A(EVEN)=',1X,G20.10)
373
374      C   HERE THE PROGRAM DETERMINES WHICH CHARACTERISTIC NUMBER THAT
375      C   AGUESS IS CLOSER TO, ADDD OR AEVEN, AND THEN WRITING INTO
376      C   MATHIEUMUN THE RESULT, USING THIS VALUE FOR CALCULATING THE
377      C   SOLUTIONS.
378
379      FACT-FACTOR FOR DETERMINING SOLUTION PERIOD (1.0 IS FOR EVEN ORDER
380      C   AND 2.0 FOR ODD)
381
382      C
383      IF(ADDD.LE.AEVEN) GO TO 380
384      IF(AGUESS.LE.AODD) GO TO 380
385      GO TO 400
386      380 CONTINUE
387      IF(AGUESS.LE.AEVEN) GO TO 420
388      DIFF1=ADDD-AGUESS
389      DIFF2=AGUESS-AEVEN
390      IF(DIFF1.GE.DIFF2) GO TO 440
391      GO TO 420
392
393      380 CONTINUE
394      IF(AGUESS.LE.AEVEN) GO TO 400
395      GO TO 420
396      400 CONTINUE
397      IF(AGUESS.LE.AODD) GO TO 440
398      DIFF1=AEVEN-AGUESS
399      DIFF2=AGUESS-AODD
400      IF(DIFF1.GT.DIFF2) GO TO 440
401      GO TO 420
402      410 CONTINUE
403      IF(IPAR.EQ.1) GO TO 440
404
405      420 CONTINUE
406      A=AEVEN
407      IPAR=1
408      FACT=1.0
409      WRITE(10,430) A
410      430 FORMAT(10,'CHARACTERISTIC NUMBER (EVEN ORDER)=',1X,G20.10)
411      GO TO 480
412      440 CONTINUE
413      A=ADDD
414      FACT=2.0
415      IPAR=0
416      WRITE(10,450) A
417      450 FORMAT(10,'CHARACTERISTIC NUMBER (ODD ORDER)=',1X,G20.10)
418
419      C
420      C   STEPS 480 TO 500 ARE THE PIECEWISE LINEAR SOLUTION.
421
422      C
423      C   AMPLP=ARRAY FOR PIECEWISE LINEAR DISPLACEMENT
424      C   SIGDP=LARGEST VALUE FOR AMPLP (TO BE USED FOR PLOT SCALING)
425      C   Z=ARRAY FOR INSTANTANEOUS NORMALIZED VALUES FOR Z
426      C   X0=INITIAL DISPLACEMENT OF LINEAR SEGMENT
427      C   XPO=INITIAL VELOCITY OF LINE
428      C   NPOINT=NUMBER OF DATA POINTS (USED FOR PLOTTING)
429
430      480 CONTINUE
431      AMPLP(1)=DISPL0
432      SIGDP=DABS(DISPL0)
433      Z(1)=Z0/PI
434      T(1)=Z0
435      X0=DISPL0
436      XPO=VO
437      NPOINT=FACT*N
438
439      C
440      C   IK=COUNTER FOR FILLING REMAINING ELEMENTS OF AMPLP AND Z

```

POOR COPY
COPIE DE QUALITEE INFERIEURE

```

441 C      PSQ=SQUARE OF ANGULAR FREQUENCY OF APPROXIMATED
442 C      X=FINAL DISPLACEMENT OF LINEAR SEGMENT
443 C      XP=FINAL LINE VELOCITY
444 C
445 C
446 C      IF PSQ IS POSITIVE, THE SOLUTION IS OF THE FAMILIAR TRIGONOMETRIC
447 C      FORM. IF ZERO, THE SOLUTION IS LINEAR (STEP 470) AND IF NEGATIVE,
448 C      HYPERBOLIC (STEP 480).
449 C
450 C#####
451 DO 800 I=1,NPOINT
452 IP=I+1
453 PSQ=2.0*P*DCOS((2.0*DFLDAT(I))-1.0)*DELTAZ
454 IF(PSQ.LT.-1.0D-10) GO TO 480
455 IF(DABS(PSQ).LE.1.0D-10) GO TO 470
456 PSOSORT(PSQ)
457 X=XO-XCOS(P*DELTAZ)*(XP0/P)+DSIN(P*DELTAZ)
458 XP=XP0+DSIN(P*DELTAZ)+XP0*DCOSH(P*DELTAZ)
459 GO TO 480
460 CONTINUE
461 X=XP0*DELTAZ+X0
462 XP=XP0
463 GO TO 480
464 480 CONTINUE
465 PSOSORT(-PSQ)
466 X=XO-DCOSH(P*DELTAZ)*(XP0/P)-DSINH(P*DELTAZ)
467 XP=XP0+DSINH(P*DELTAZ)+XP0*DCOSH(P*DELTAZ)
468 480 CONTINUE
469 AMPLP(IP)=K
470 Z(IP)=DFLDAT(I)*DELTAZ/PI
471 C#####
472 C      T=DOUBLE-PRECISION ARRAY FOR Z (TO BE USED FOR FINDING SERIES
473 C      APPROXIMATION)
474 C
475 C#####
476 T(IP)=DFLDAT(I)*DELTAZ
477 X0=X
478 XP=XP
479 IF(DABS(X).GT.BIGP) BIGP=DABS(X)
480 800 CONTINUE
481 C#####
482 C      THE FOLLOWING CALCULATES THE COEFFICIENTS OF THE SERIES
483 C      APPROXIMATION. USING THE METHOD DESCRIBED BY E.L. INCE IN
484 C      "TABLES OF THE ELLIPTIC CYLINDER FUNCTIONS" (PROC. G.Y. SOC.
485 C      EDINBURGH, VOL. 82, P. 366-423, 1932) AND N.W. MCLACHLAN IN
486 C      "THEORY AND APPLICATION OF MATHEU FUNCTIONS", CH. 3, P. 28 FF.,
487 C      OXFORD UNIVERSITY PRESS, 1947
488 C
489 C      IR1= COUNTER FOR CALCULATING PRELIMINARY COEFFICIENT RATIOS
490 C      GAMMA=FACTOR USED FOR CALCULATING CONTINUED FRACTIONS USED FOR
491 C      ABOVE
492 C
493 C      FOR A NON-ZERO INITIAL DISPLACEMENT AND AN EVEN ORDER
494 C      CHARACTERISTIC NUMBER, THE PROGRAM MOVES TO STEP 510. FOR A ZERO
495 C      INITIAL DISPLACEMENT AND EVEN ORDER, THE CALCULATIONS START AT
496 C      STEP 570. FOR NON-ZERO INITIAL DISPLACEMENT AND ODD ORDER, THE
497 C      ITERATIONS BEGIN AT STEP 580. FOR 000 ORDER AND ZERO
498 C      DISPLACEMENT, STEP 600 IS WHERE THE PROGRAM CONTINUES.
499 C
500 C#####
501 S02
502 IR1=26
503 GAMMA=0.0
504 IF((DISPLO.NE.0.0).AND.(IPAR.EQ.1)) GO TO 510
505 IF((DISPLO.EQ.0.0).AND.(IPAR.EQ.1)) GO TO 570
506 IF((DISPLO.NE.0.0).AND.(IPAR.EQ.0)) GO TO 580
507 GO TO 600
508 C#####
509 S10
510 S11
511 C      THE CASE OF NON-ZERO INITIAL DISPLACEMENT AND EVEN ORDER.
512 C
513 C      RATIO=RATIO BETWEEN COEFFICIENT FOR SPECIFIC TERM AND ONE FOR
514 C      PREVIOUS TERM
515 C      SUB-TERM USED IN SUMMING SQUARES OF VARIOUS RATIO VALUES
516 C      IEND=COUNTER USED IN CALCULATING VARIOUS VALUES FOR RATIO
517 C
518 C#####
519 S10 CONTINUE
520 RATIO(1)=A/0
521 RATIO(2)=(A-4.0)/0-2.0/RATIO(1)
522 SUM=2.0
523 IEND=2
524 C#####
525 C      THE CONTINUED FRACTION FOR CALCULATING RATIO.
526 C
527 C      I1STAR=COUNTER USED FOR RATIO1
528 C
529 C#####
530 S20 CONTINUE
531 I1STAR=IR1
532 IF((DISPLO.EQ.0.0).AND.(IPAR.EQ.1)) I1STAR=IR1-1
533 RATIO1(I1STAR)=0/(A-(2.0*DFLOAT(IR1))**2-0-GAMMA)
534 GAMMARATIO1(I1STAR)
535 IR1=IR1-1
536 IF(IR1.EQ.IEND) GO TO 530
537 GO TO 520
538 C#####
539 S40
540 C      RATIO2=RATIO BETWEEN COEFFICIENT FOR SPECIFIC TERM AND COEFFICIENT
541 C      FOR PREVIOUS ONE
542 C      IR2,I2STAR=COUNTERS FOR RATE02
543 C
544 C#####
545 S30 CONTINUE
546 RATIO2(1)=RATIO1(1)
547 DO 540 IR2=1,23
548 I2STAR=IR2+1
549 RATIO2(I2STAR)=RATIO2(IR2)*RATIO1(I2STAR)
550

```

```

681      640 CONTINUE
682      C#####
683      C      BY SUMMING ALL THE VARIOUS VALUES FOR RATIO2, AND INVERTING THE
684      C      SQUARE ROOT, THE VALUE FOR THE FIRST COEFFICIENT IS DETERMINED AND
685      C      SUBSEQUENTLY, ALL THE REMAINING ONES.
686      C      COEFFARRAY FOR COEFFICIENT VALUES
687      C      IC1,IC2=COUNTERS FOR COEFF
688      C
689      C#####
690      IR1=24
691      DO 650 I=1,IR1
692      SUMSUM=RATIO2(I)=0
693      650 CONTINUE
694      COEFF(1)=1.0/DSQRT(SUM)
695      DO 650 IC1=1,IR1
696      IC2=IC1+
697      COEFF(IC2)=COEFF(1)=RATIO2(IC1)
698      650 CONTINUE
699      GO TO 610
700      C#####
701      C      2 INITIALIZING THE CALCULATIONS FOR THE SITUATION OF ZERO INITIAL
702      C      AND EVEN ORDER.
703      C
704      C#####
705      670 CONTINUE
706      RATIO1(1)=(A-4.0)/0
707      SUM=1.0
708      IEND=1
709      IR1=28
710      DO 620
711      C#####
712      C      SETTING UP THE CASE OF NON-ZERO DISPLACEMENT AND ODD ORDER.
713      C
714      C#####
715      680 CONTINUE
716      RATIO1(1)=(A-1.0-0)/0
717      SUM=1.0
718      IEND=1
719      IR1=IR1-1
720      IF(IR1.EQ.IEND) GO TO 630
721      GO TO 680
722      C#####
723      C      THE SITUATION OF ZERO INITIAL DISPLACEMENT AND ODD ORDER.
724      C
725      C#####
726      690 CONTINUE
727      RATIO1(1)=(A-1.0+0)/0
728      SUM=1.0
729      IEND=1
730      GO TO 680
731      C#####
732      C      THE SECTION FROM STEP 610 TO STEP 650 IS THE SERIES APPROXIMATION
733      C      THAT IS DESCRIBED IN INCE AND McLACHLAN.
734      C
735      C      BIGS=LARGEST VALUE FOR SERIES AMPLITUDE (FOR PLOT SCALING)
736      C
737      C#####
738      610 CONTINUE
739      BIGS=DABS(DISPL0)
740      MPOINT=FACT+H+1.0
741      C#####
742      C      INITIALIZING THE CALCULATIONS OF THE SERIES TERMS.
743      C
744      C      DENOM=SUMMATION OF SERIES COEFFICIENTS (FOR NON-ZERO INITIAL
745      C      DISPLACEMENTS) OR COEFFICIENTS AND TERM NUMBER (FOR ZERO
746      C      INITIAL DISPLACEMENTS) USED IN DETERMINING MATHEU FUNCTION
747      C      COEFFICIENT FOR COMPLETE SOLUTION.
748      C      TERM=SUM OF INDIVIDUAL TERMS FOR MATHEU FUNCTION APPROXIMATION
749      C
750      C#####
751      DO 650 J=1,MPOINT
752      DENOM=0.0
753      TERM=0.0
754      IF(DISPL0.EQ.0.0) GO TO 630
755      C#####
756      C      RESULTS FOR NON-ZERO INITIAL DISPLACEMENT.
757      C
758      C      THETA=TERM NUMBER
759      C
760      C#####
761      DO 620 K=1,25
762      IF(IPAR.EQ.0) THETA=2.0*DFLOAT(K)-1.0
763      IF(IPAR.EQ.1) THETA=2.0*DFLOAT(K)-2.0
764      DENOM=DENOM+COEFF(K)
765      TERM=TERM+COEFF(K)*DCOS(THETA*T(J))
766      620 CONTINUE
767      C#####
768      C      AMPLS=AMPLITUDE ARRAY FOR INSTANTANEOUS AMPLITUDE FOR SERIES SOLUTION
769      C      DISPLS=DOUBLE-PRECISION VALUE OF AMPLS.
770      C
771      C#####
772      AMPLS(J)=DISPL0*TERM/DENOM
773      DISPLS=DOUBLE(AMPLS(J))
774      IF(DABS(DISPLS).GT.BIGS) BIGS=DABS(DISPLS)
775      GO TO 650
776      C#####
777      630 CONTINUE
778      C#####
779      660 C

```

```

661 C      CALCULATIONS FOR ZERO INITIAL DISPLACEMENT.
662 C
663 C#####
664 DO 640 K=1,25
665 IF(IPAR.EQ.0) THETA=2.0*DFLOAT(K)-1.0
666 IF(IPAR.EQ.-1) THETA=2.0*DFLOAT(K)
667 DENOM=2.0*THETA+Coeff(K)
668 TERM=TERM+Coeff(K)*DSIN(THETA*T(J))
669 640 CONTINUE
670 AMPLS(J)=VO*TERM/DENOM
671 DISPLS=2.0*(AMPLS(J))
672 IF(DABS(DISPLS).GT.BIGS) BIGS=DABS(DISPLS)
673 680 CONTINUE
674 C#####
675 C      THE SECTION ENDING AT STEP 680 IS THE RUNGE-KUTTA SOLUTION.
676 C
677 C
678 C
679 C      THE FOLLOWING ARE PARAMETERS FOR THE RUNGE-KUTTA ROUTINE.
680 C      CONSULT THE IMSL MANUAL ON "DVERK" FOR FURTHER DETAILS.
681 C
682 C#####
683 NPOINT=FACTAH+1.0
684 SIGRK=DAbs(DISPL0)
685 Y(1)=DISPL0
686 Y(2)=VO
687 ZRK=2.0
688 NCON=2
689 NW=2
690 TOL=1.0D-12
691 IND=1
692 ZEND=ZEND+DELTAZ
693 DO 680 K=1,NPOINT
694 AMPLRK(K)=Y(1)
695 CALL DVERK(NCON,FCN,ZRK,Y,ZEND,TOL,IND,C,NW,W,IER)
696 C#####
697 C      INCREMENTING ZEND.
698 C
699 C#####
700 ZEND=ZEND+DELTAZ
701 IF(BABS(Y(1)).GT.SIGRK) SIGRK=Y(1)
702 680 CONTINUE
703 C#####
704 C      WRITING THE SOLUTIONS INTO OUTPUT FILE MATHIEUMUM.
705 C
706 C#####
707 WRITE(10,870)
708 870 FORMAT('0',1X,'2',1X,'PIECEWISE LINEARIZATION',2X,'SERIES APPROX'
709 'IMATION',2X,'RUNGE-KUTTA')
710 DO 680 L=1,NPOINT
711 WRITE(10,680) Z(L),AMPLP(L),AMPLS(L),AMPLRK(L)
712 680 FORMAT(1X,A(4(G20.10,2X)))
713 680 CONTINUE
714 C#####
715 C      THE NEXT FEW LINES ARE USED IN FINALIZING ALL THE PLOT PARAMETERS.
716 C      WHICH INCLUDES AXIS SCALING.
717 C
718 C#####
719 INC=M
720 XA(2)=Z(NPOINT)-0.005
721 SIGRKSIGRK
722 IF((SIGP.GT.BIGS).AND.(SIGP.GT.SIGRK)) SIG=SIGP
723 IF((BIGS.GT.SIGP).AND.(SIGS.GT.SIGRK)) SIG=BIGS
724 M=1
725 IF(SIG.LE.1.0) GO TO 710
726 700 CONTINUE
727 DEC=BIG/(10.0**M)
728 IF(DEC.LE.1.0) GO TO 730
729 M=M+1
730 GO TO 700
731 710 CONTINUE
732 DEC=BIG/10.0
733 M=M+1
734 GO TO 710
735 720 CONTINUE
736 DEC=BIG/10.0
737 M=(M-1)
738 GO TO 720
739 740 CONTINUE
740 DEC=BIG/10.0
741 M=(M-1)
742 750 CONTINUE
743 IF(DEC.LE.0.125) VFACT=0.125
744 IF((DEC.GT.0.125).AND.(DEC.LE.0.25)) VFACT=0.25
745 IF((DEC.GT.0.25).AND.(DEC.LE.0.5)) VFACT=0.5
746 IF((DEC.GT.0.5).AND.(DEC.LE.1.0)) VFACT=1.0
747 SFACT=VC/10.0
748 VAH=VFACT*10.0**M
749 DEN=VC/(2.0*SFACT)
750 VB=(VFACT/DEN)*10.0**M
751 HB=Z(NPOINT)/DFLOAT(INC)
752 C#####
753 C      THE FOLLOWING SECTION SETS UP THE PIECEWISE LINEAR AND RUNGE-KUTTA
754 C      RESULTS SO THAT 25 POINTS OF THE PIECEWISE SOLUTION AND 27 OF THE
755 C      RUNGE-KUTTA VALUES ARE PLOTTED FOR A CLEARER REPRESENTATION, WITH
756 C      THE PLOTS BEING ALTERNATED FOR COMPARISON PURPOSES.
757 C
758 C      YP=ARRAY FOR PLOTTED PIECEWISE LINEAR DISPLACEMENTS
759 C      ZP=ARRAY FOR PLOTTED PIECEWISE LINEAR Z VALUES
760 C      YRK=ARRAY FOR PLOTTED RUNGE-KUTTA DISPLACEMENTS
761 C      ZPRK=ARRAY FOR PLOTTED RUNGE-KUTTA Z VALUES
762 C
763 C
764 C      **NOTE** THE NUMBER OF POINTS TO BE PLOTTED MUST NOT EXCEED 1000.
765 C
766 C#####
767 INCR=NPOINT/25
768 INCRRK=INCR/2

```

```

771      NPP=1
772      NPK=1
773      NDP=1
774      NDK=1
775      740 CONTINUE
776          YP(NPP)=AMPLR(NDP),
777          ZP(NPP)=Z(NDP)
778          IF(NDP.EQ.NPOINT) GO TO 750
779          NPP=NPP+1
780          NDP=NDP+INCR
781          GO TO 740
782      750 CONTINUE
783          YRK(NPRK)=AMPLR(NDK)
784          ZPRK(NPRK)=Z(NDK)
785          IF(NDK.EQ.1) GO TO 760
786          NDK=NDK+INCR
787          GO TO 770
788      760 CONTINUE
789          NDK=NDK+INCR
790      770 CONTINUE
791          IF(NDK.GT.NPOINT) GO TO 780
792          NPK=NPK+1
793          GO TO 760
794      780 CONTINUE
795          NPK=NPK+1
796          YRK(NPRK)=AMPLR(NPOINT)
797          ZPRK(NPRK)=Z(NPOINT)
798
799 C PLOTTING THE SOLUTIONS. CONSULT THE WRITEUP ON CGPL/CGPL2 AND
800 C THE MANUAL ON DIGITAL PLOTTING FOR DETAILS.
801 C
802 C
803 C
804      NF=1
805      ND=NPOINT
806      CALL CGPL2(Z,AMPLS,ND,NF,1,HA,HB,HC,VA,VB,VC,ALPH)
807      ND=NPP
808      CALL CGPL2(ZP,YP,ND,2,1,HA,HB,HC,VA,VB,VC,ALPH)
809      ND=NPRK
810      CALL CGPL2(ZPRK,YRK,ND,3,1,HA,HB,HC,VA,VB,VC,ALPH)
811      NF=4
812      CALL CGPL2(XA,YA,2,NF,4,HA,HB,HC,VA,VB,VC,ALPH)
813      HORIZ=4.15
814      VERT=-0.475
815      CALL SYMBOL(HORIZ,VERT,0,2,183,0,0,-1)
816      HORIZ=HA
817      VERT=VC+0.5
818      CALL PLOT(HORIZ,VERT,3)
819      HORIZ=HC
820      CALL PLOT(HORIZ,VERT,2)
821      VERT=VERT+4.0
822      CALL PLOT(HORIZ,VERT,2)
823      HORIZ=HA
824      CALL PLOT(HORIZ,VERT,2)
825      VERT=VC+0.5
826      CALL PLOT(HORIZ,VERT,2)
827      STARTX=(HC-7.7)/2.0
828      STARTY=VC+0.5
829      HORIZ=STARTX+0.2
830      VERT=STARTY+3.6
831      CALL SYMBOL(HORIZ,VERT,0,2,'CHAR. NUM. (EST. VALUE). #',0,0,25)
832      HORIZ=STARTX+5.6
833      CALL NUMBER(HORIZ,VERT,0,2,AQUES,0,0,4)
834      HORIZ=STARTX+0.2
835      VERT=STARTY+3.2
836      CALL SYMBOL(HORIZ,VERT,0,2,'CHAR. NUM. (FINAL VALUE). #',0,0,28)
837      HORIZ=STARTX+5.6
838      CALL NUMBER(HORIZ,VERT,0,2,B,0,0,4)
839      HORIZ=STARTX+0.2
840      VERT=STARTY+2.8
841      CALL SYMBOL(HORIZ,VERT,0,2,'0 #',0,0,3)
842      HORIZ=STARTX+1.0
843      CALL NUMBER(HORIZ,VERT,0,2,0,0,4)
844      HORIZ=STARTX+3.6
845      CALL SYMBOL(HORIZ,VERT,0,2,E2,0,0,-1)
846      HORIZ=STARTX+3.7
847      CALL SYMBOL(HORIZ,VERT,0,2,'2 #',0,0,3)
848      HORIZ=STARTX+4.6
849      CALL SYMBOL(HORIZ,VERT,0,2,183,0,0,-1)
850      HORIZ=STARTX+4.7
851      CALL SYMBOL(HORIZ,VERT,0,2,B7,0,0,-1)
852      HORIZ=STARTX+4.9
853      CALL NUMBER(HORIZ,VERT,0,2,M,0,0,1)
854      HORIZ=STARTX+0.2
855      VERT=STARTY+2.0
856      CALL SYMBOL(HORIZ,VERT,0,2,'Y(0) #',0,0,8)
857      HORIZ=STARTX+1.6
858      CALL NUMBER(HORIZ,VERT,0,2,DISPL0,0,0,4)
859      HORIZ=STARTX+3.6
860      CALL SYMBOL(HORIZ,VERT,0,2,'Y',0,0,1)
861      HORIZ=STARTX+3.7
862      CALL SYMBOL(HORIZ,VERT,0,2,125,0,0,-1)
863      HORIZ=STARTX+3.9
864      CALL SYMBOL(HORIZ,VERT,0,2,'(0) #',0,0,5)
865      HORIZ=STARTX+5.1
866      CALL NUMBER(HORIZ,VERT,0,2,Y0,0,0,4)
867      HORIZ=STARTX+0.3
868      VERT=STARTY+1.1
869      CALL PLOT(HORIZ,VERT,3)
870      HORIZ=STARTX+0.8
871      CALL PLOT(HORIZ,VERT,2)
872      HORIZ=STARTX+1.0
873      VERT=STARTY+1.0
874      CALL SYMBOL(HORIZ,VERT,0,2,'SERIES APPROXIMATION',0,0,20)
875      HORIZ=STARTX+0.6
876      VERT=STARTY+0.7
877      CALL SYMBOL(HORIZ,VERT,0,2,I,0,0,-1)
878      HORIZ=STARTX+1.0
879      VERT=STARTY+0.6
880      CALL SYMBOL(HORIZ,VERT,0,2,'PIECEWISE LINEARIZATION',0,0,23)

```

```

881      HORIZ=STARTX+0.8
882      VERT=STARTY+0.3
883      CALL SYMBOL(HORIZ,VERT,0.2,2,0.0,-1)
884      HORIZ=STARTX+1.0
885      VERT=STARTY+0.2
886      CALL SYMBOL(HORIZ,VERT,0.2,'RUNGE-KUTTA',0.0,11)
887      NF#0
888      CALL CGPL2(XA,YA,2,NF,4,HA,HB,HC,VA,VB,VC,ALPH)
889      STOP
890      END
891      C
892      C
893      C
894      C
895      C
896      C
897      C
898      C
899      C
900      C
901      C#####
902      C
903      C      THE FOLLOWING SUBROUTINE CALCULATES THE OUTPUT VALUE FOR A GUESS
904      C      FOR THE CHARACTERISTIC NUMBER, USING THE RECURRANCE RELATIONS THAT
905      C      ARE DESCRIBED IN MCLACHLAN AND INCE.
906      C
907      C#####
908      SUBROUTINE RECUR
909      IMPLICIT REAL*8 (A-N,D-Z)
910      COMMON//CALC/AIN,DISPL0,DIFF,AOUT,IPAR
911      COMMON//PARAM/Q
912      C#####
913      C      'RE-COUNTER FOR THE NUMBER OF ITERATIONS FOR THE PARTICULAR RELATION
914      C      USED
915      C      GAMMAR=FACTOR USED IN THE CONTINUED FRACTION (SEE INCE).
916      C
917      C      FOR NON-ZERO INITIAL DISPLACEMENT AND EVEN ORDER, THE PROGRAM
918      C      MOVES TO STEP 800. FOR ZERO INITIAL DISPLACEMENT AND EVEN ORDER,
919      C      IT GOES TO STEP 820. IF THE ORDER IS ODD AND INITIAL DISPLACEMENT
920      C      NON-ZERO, IT CONTINUES AT STEP 840, AND IF THE DISPLACEMENT
921      C      HAPPENS TO BE ZERO, IT GOES TO STEP 860.
922      C
923      C#####
924      R1200.0
925      GAMMAR=0.0
926      IF((DISPL0.NE.0.0).AND.(IPAR.EQ.1)) GO TO 800
927      IF((DISPL0.EQ.0.0).AND.(IPAR.EQ.1)) GO TO 820
928      IF((DISPL0.NE.0.0).AND.(IPAR.EQ.0)) GO TO 840
929      GO TO 860
930      C#####
931      C
932      C      THE CASE OF NON-ZERO INITIAL DISPLACEMENT AND EVEN ORDER.
933      C
934      C      PHIR, DENOMR=FACTORS USED IN CALCULATING CONTINUED FRACTION
935      C      A=VALUE OBTAINED
936      C#####
937      C
938      800 CONTINUE
939      PHIR=1.0-AIN/(4.0*R**2)-GAMMAR
940      DENOMR=(Q**2)/((16.0*R**2)*(R-1.0)**2)
941      GAMMAR=DENOMR/PHIR
942      R=R-1.0
943      IF(R.EQ.1.0) GO TO 810
944      GO TO 800
945      810 CONTINUE
946      A=(Q**2)/(2.0*(1.0-AIN/4.0-GAMMAR))
947      GO TO 870
948      C#####
949      C
950      820 CONTINUE
951      C      ZERO INITIAL DISPLACEMENT AND EVEN ORDER.
952      C#####
953      C
954      830 CONTINUE
955      PHIR=1.0-AIN/(4.0*R**2)-GAMMAR
956      DENOMR=(Q**2)/((16.0*R**2)*(R-1.0)**2)
957      GAMMAR=DENOMR/PHIR
958      R=R-1.0
959      IF(R.EQ.2.0) GO TO 830
960      GO TO 820
961      830 CONTINUE
962      A=4.0-(Q**2)/(16.0*(1.0-AIN/16.0-GAMMAR))
963      GO TO 870
964      C#####
965      C
966      C      THE SITUATION OF ODD ORDER AND NON-ZERO INITIAL DISPLACEMENT.
967      C
968      840 CONTINUE
969      PHIR=1.0-AIN/(2.0*R+1.0)**2-GAMMAR
970      DENOMR=(Q**2)/((14.0*R**2-1.0)**2)
971      GAMMAR=DENOMR/PHIR
972      IF(R.EQ.1.0) GO TO 850
973      R=R+1.0
974      GO TO 840
975      850 CONTINUE
976      A=1.0+Q-GAMMAR
977      IF(0.LT.Q.0) Q=DABS(Q)
978      GO TO 870
979      C#####
980      C
981      C      THE CASE OF ODD ORDER AND ZERO INITIAL DISPLACEMENT
982      C      Q=0
983      C
984      860 CONTINUE
985      Q=0
986      GO TO 840
987      870 CONTINUE
988      AOUT=A
989      C#####
990      C

```


VITA

NAME: Bernhard Michael Jatzeck

PLACE OF BIRTH: Berlin Spandau, Federal Republic of Germany

DATE OF BIRTH: October 12, 1955

POST-SECONDARY EDUCATION AND DEGREES

Faculty of Engineering, Camrose Lutheran College
Camrose, Alberta
1973-1974

University of Alberta
Edmonton, Alberta
1974-1977 B. Sc. (Mec. E.)

Dept. of Mechanical Engineering, University of British Columbia
Vancouver, B. C.
1979-1980 (Started thesis under Dr. V. J. Modi. Transferred course credits to University of Alberta).

Dept. of Mechanical Engineering, University of Alberta
Edmonton, Alberta
1980-1981 (Full-time), 1981-1982 (Part-time)

PROFESSIONAL EXPERIENCE

Junior Plant Engineer
Amoco Canada Petroleum Co. Ltd.
West Whitecourt Plant
June, 1977 - December, 1977

Process Engineer
Amoco Canada Petroleum Co. Ltd.
Crossfield Area Office
Calgary, Alberta
January, 1978 - April, 1978

Junior Project Engineer
Ingersoll-Rand Canada Inc.
Packaged Equipment Division
Calgary, Alberta
May, 1978 - August, 1979

PROFESSIONAL EXPERIENCE (continued)

Mechanical Engineer

SED Systems Inc

Advanced Systems Engineering Group (Aerospace Products)

Saskatoon, Saskatchewan

November, 1981 - Present

PROFESSIONAL SOCIETIES

Registered Professional Engineer

Association of Professional Engineers, Geologists, and Geophysicists of Alberta

Registered Professional Engineer

Association of Professional Engineers of Saskatchewan

Member

Canadian Society for Mechanical Engineering

Member

Engineering Institute of Canada

New Member

American Institute of Aeronautics and Astronautics

Student Member

American Association for the Advancement of Science (to be re-instated as full member upon renewal)