

AD-A258 970



AFIT/GA/ENY/92D-03

①

CANONICAL FLOQUET PERTURBATION THEORY

THESIS

David J. Pohlen, Captain, USAF

AFIT/GA/ENY/92D-03

DTIC
S ELECTE D
E
JAN 06 1993

Approved for public release; Distribution Unlimited

93-00130

162/28

93 1 04 012

CANONICAL FLOQUET PERTURBATION THEORY

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Astronautical Engineering

David J. Pohlen, B.S.

Captain, USAF

December 1992

DTIC QUALITY INSPECTED 8

Accession For	
NTIS CRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification _____	
By _____	
Distribution / _____	
Availability Codes	
Dist	Avail and/or Special
A-1	

Approved for public release; Distribution Unlimited

Acknowledgments

I would like to thank my advisor, Dr. William Wiesel for all of his work, help, and contributions to this study. Due to the limited knowledge base on this subject, I very likely would not have completed near the amount of work presented here, without Dr. Wiesel's vital assistance.

A special thanks also to Captains Dan Uribe and Janice Horn who's assistance in explaining concepts and in understanding quirks of Fortran code was invaluable in saving me many hours of wasted effort.

In dedicating this work, I would like to acknowledge several people. I would first like to thank my parents, for without their love and support (and sometimes pushing and shoving) for the first twenty odd years of my life, a college education, an engineering degree, and a career in the Air Force would not have become a reality. Secondly, I would like to thank my grandparents, because through their average 90 plus years of life, they have shown me that its not only the big events that make life important, but all the small events which add the flavor and enjoyment to life. Finally and by no means least, I want to thank Amy my wife of five years, who has given a lot of herself, for me, for my education, for my career, and most importantly for our family. With all my love, I dedicate this work to you Amy.

- David J Pohlen

Table of Contents

	page
Acknowledgments	ii
List of Figures	v
List of Symbols	viii
Abstract	xi
I. Introduction	1
II. Historical Development	3
2.1 Canonical Transformations	3
2.2 Real Canonical Transformations	3
2.3 Perturbation Theory on Periodic Orbits	4
III. Theory	5
3.1 Classical Floquet Theory	5
3.2 Canonical Floquet Theory	9
3.3 Real Valued Symplectic Eigenvectors	17
3.4 Modal Variables and E After Real Canonical Transformation	21
3.5 The Restricted Three-Body Problem	22
3.6 Perturbation Theory on the Restricted Three-Body Problem	30
IV. Software	40
4.1 Surface of Section	40
4.2 Determination of Periodic Initial Conditions	40
4.3 Symplectic Normalization	41
4.4 Storage of the Periodic Trajectory and Hamiltonian Coefficients	42
4.5 The Perturbation Solutions	43
V. Results and Discussions	45
5.1 Canonical Floquet Theory	45
5.2 The Three-Body Perturbation Problem	45
5.3 Summary	84
VI. Conclusions and Recommendations	91

Appendix A: Six Modal Equation of Variation Types .	93
Appendix B: Fortran Code	100
Bibliography	147
Vita	148

List of Figures

Figure	Page
1. Reference Frame for the Restricted Three-body System	25
2. An Elliptical Trajectory Precessing About $1-\mu$.	31
3. The Surface of Section of the Elliptical Trajectory	32
4. The Surface of Section for the Sun-Jupiter System	33
5. Close-up View of Periodic Region of the Sun-Jupiter Surface of Section	47
6. Unperturbed Oscillatory Modal Variables, y_1 versus y_2	49
7. Unperturbed Time History of Modal Variable y_3 . .	50
8. Unperturbed Time History of Modal Variable y_4 . .	51
9. Oscillatory Modal Variables, y_1 versus y_2 , $\delta q_1 = 1e-8$	53
10. Time History of Modal Variable y_1 , $\delta q_1 = 1e-8$.	54
11. Time History of Modal Variable y_2 , $\delta q_1 = 1e-8$.	55
12. Oscillatory Modal Variables, y_1 versus y_2 , $\delta q_1 = 1e-6$	57
13. Time History of Modal Variable y_3 , $\delta q_1 = 1e-6$.	58
14. Time History of Modal Variable y_4 , $\delta q_1 = 1e-6$.	59
15. Oscillatory Modal Variables, y_1 versus y_2 , $\delta q_1 = 5e-5$	60
16. Oscillatory Modal Variables, y_1 versus y_2 , Expanded Solution, $\delta q_1 = 5e-5$	61
17. Time History of Modal Variable y_1 , Exact Solution, $\delta q_1 = 5e-5$	63
18. Time History of Modal Variable y_1 , Expanded Solution, $\delta q_1 = 5e-5$	64

19.	Time History of Modal Variable y_2 , $\delta q_1 = 5e-5$	65
20.	Time History of Modal Variable y_3 , $\delta q_1 = 5e-5$	66
21.	Time History of Modal Variable y_4 , $\delta q_1 = 5e-5$	67
22.	Oscillatory Modal Variables, y_1 versus y_2 , $\delta q_1 = 1e-4$	68
23.	Oscillatory Modal Variables, y_1 versus y_2 , Exact Solution, $\delta q_1 = 1e-3$	69
24.	Time History of Modal Variable y_1 , Exact Solution, $\delta q_1 = 1e-3$	70
25.	Time History of Modal Variable y_2 , Exact Solution, $\delta q_1 = 1e-3$	71
26.	Oscillatory Modal Variables y_1 versus y_2 , $\delta J = 1e-8$	73
27.	Oscillatory Modal Variables y_1 versus y_2 , $\delta J = 1e-6$	74
28.	Time History of Modal Variable y_1 , $\delta J = 1e-6$	75
29.	Time History of Modal Variable y_2 , $\delta J = 1e-6$	76
30.	Time History of Modal Variable y_3 , $\delta J = 1e-6$	77
31.	Time History of Modal Variable y_4 , $\delta J = 1e-6$	78
32.	Oscillatory Modal Variables y_1 versus y_2 , $\delta J = 1e-5$	80
33.	Oscillatory Modal Variables y_1 versus y_2 , $\delta J = 5e-5$	81
34.	Time History of Modal Variable y_1 , $\delta J = 1e-5$	82
35.	Time History of Modal Variable y_1 , Exact Solution, $\delta J = 5e-5$	83
36.	Oscillatory Modal Variables y_1 versus y_2 , Exact Solution, $\delta J = 1e-3$	85
37.	Time History of Modal Variable y_1 , Exact Solution $\delta J = 1e-3$	86
38.	Time History of Modal Variable y_2 , Exact Solution $\delta J = 1e-3$	87

39.	Time History of Modal Variable y_3 , Exact Solution $\delta J = 1e-3$	88
40.	Time History of Modal Variable y_4 , Exact Solution $\delta J = 1e-3$	89

List of Symbols

$\mathbf{A}(t)$	Linearization of Dynamics
c_1	Expansion Coefficient
$\mathbf{D}$	Diagonal Matrix of Symplectic Normalization Multipliers
d_1	Symplectic Normalization Multiplier
$\mathbf{E}$	Matrix of Symplectic Eigenvectors
$\mathbf{E}'$	$\mathbf{E}$ Matrix After Type A Transformation
$\mathbf{E}''$	$\mathbf{E}$ Matrix After Type B Transformation
$\mathbf{F}(t)$	System Eigenvector Matrix
H	System Hamiltonian
H_1	First Order Partial of Hamiltonian
H_{1j}	Second Order Partial of Hamiltonian
H_{1jk}	Third Order Partial of Hamiltonian
$\mathcal{H}$	Variational Hamiltonian
$\mathbf{I}$	Identity Matrix
$\mathbf{J}$	Jordan Normal Matrix of Poincaré Exponents
$\mathbf{J}'$	$\mathbf{J}$ Matrix After Type A Transformation
$\mathbf{J}''$	$\mathbf{J}$ Matrix After Type B Transformation
J	Jacobian Constant
δJ	Change in Jacobian Constant
K	New Hamiltonian After Canonical Transformation
$\mathcal{K}$	New Variational Hamiltonian After Transformation
M_1	Mass of Body
m_1	Dimensionless Mass of Body

p_1	Hamiltonian Momenta, Conjugate to Coordinate q_1
δp_1	Change in Momenta
q_1	Hamiltonian Coordinate Variable
δq_1	Change in Coordinate
r_1	Position Vector of Primary Mass
S	$\partial^2 K / \partial \bar{Y}^2$
S'	S Matrix After Type A Transformation
S''	S Matrix After Type B Transformation
S_1	Distance of Primary Body to System Center of Mass
s_1	Dimensionless Distance of Primary Body to System Center of Mass
T	Transformation Matrix
T_A	Type A Transformation Matrix - Eliminates Imaginary Poincaré
T_B	Type B Transformation Matrix - Eliminates Imaginary Symplectic Eigenvectors
$\bar{X}$	System State Vector
$\bar{x}$	State Vector for Exact Minus Nearly Periodic Orbit
x	Coordinate in Jefferys' Equations of Motion
$\bar{Y}$	New State Vector After Canonical Transformation
$\bar{y}(t)$	Modal State Vector
y_1	Modal Variable
y	Coordinate in Jefferys' Equations of Motion
Z	Correlation Matrix for Equations of Motion
α	Arbitrary Multiple
$\bar{e}_1$	System Eigenvector
i	Imaginary Number
Λ	Matrix of Eigenvalues in Jordan Normal Form

λ_1	System Eigenvalue
μ	Dimensionless Mass and Distance Parameter
$\tilde{\rho}_1$	Symplectic Eigenvector
ρ_{1j}	The ith Row, jth Column Element of the Symplectic Eigenvector Matrix
τ	Period
$\Phi(t, t_0)$	State Transition Matrix
$\Phi(\tau+t_0, t_0)$	Monodromy Matrix
ϕ_{1j}	Element of State Transition Matrix
ω_1	Poincaré Exponent

Abstract

Classical Floquet theory is examined in order to generate a canonical transformation to modal variables for periodic systems. This transformation is considered canonical if the periodic matrix of eigenvectors is symplectic at the initial time. Approaches for symplectic normalization of the eigenvectors had to be examined for each of the different Poincaré eigenvalue cases. Particular attention was required in the degenerate case, which depended on the solution of a generalized eigenvector. Transformation techniques to ensure real modal variables and real periodic eigenvectors were also needed.

Periodic trajectories in the restricted three-body case were then evaluated using the canonical Floquet solution. The system used for analyses is the Sun-Jupiter system. This system was especially useful since it contained two of the more difficult Poincaré eigenvalue cases, the degenerate case and the imaginary eigenvalue case. The perturbation solution to the canonical modal variables was examined using both an expansion of the Hamiltonian and using a representation that was considered exact. Both methods compared quite well for small perturbations to the initial condition. As expected, the expansion solution failed first due to truncation after the third order term of the expansion.

CANONICAL FLOQUET PERTURBATION THEORY

I. Introduction

Analysis techniques of periodic systems have been available for many years since Floquet's work on time periodic linear systems in 1883. The primary use of Floquet theory is to find the characteristic or Poincaré exponents of the system and thereby determine the stability of the periodic system. In more recent studies, Floquet theory has been used to construct a set of periodic modal vectors for analysis (Wiesel, 1981; Calico and Wiesel, 1984; Ross, 1991). Unfortunately, none of these works noted that standard Floquet theory is not canonical. In order to find a useful set of periodic modal variables, a canonical version of Floquet theory must be found.

This study will look at the required methods needed to produce a canonical transformation from a time periodic linear Hamiltonian system to modal variables using Floquet theory. Special attention will be given to the various types of Poincaré exponents encountered in these types of systems.

One of the many interesting applications for canonical Floquet theory is that of periodic orbits in celestial mechanics. As is generally known, the two-body system "is

the only gravitational problem for which a closed-form solution has been found" (Wiesel, 1989:45). But in searching for exact two-body systems in our solar system alone, it is apparent that perturbations due to other bodies must be considered.

Therefore, while canonical Floquet theory is then the primary focus of this study, a secondary emphasis will be on the analysis of the restricted three-body orbit. The particular system to be looked at is the Sun-Jupiter system. While by no means the most interesting system dynamically, its mass ratio of $9.5388\text{E-}4$ makes the study easier to handle at this stage.

II. Historical Development

The initial groundwork for analysis of periodic systems was laid out by Floquet as described in Chapter I. A detailed search of the library's resources turned up very little information that expanded on the work of Floquet or would help in this study. The majority of the work found in the areas of this study is that of Dr. Wiesel and Capt Ross.

2.1 Canonical Transformations

In Ross (1991), a method for finding a set of modal vectors from a periodic system was defined, but the canonical behavior of the transformation was not adequately considered. As defined, there are actually two transformations required to change the original Hamiltonian into modal coordinates. The second, or modal transformation, was shown with great detail to be canonical (Wiesel, 1981:232-234; Pars, 1965:453-483). But, the first, or Floquet transformation, was not completely examined until Dr. Wiesel's discovery of a proof for this transformation (Siegel and Moser, 1971:99-101). This proof is presented in Chapter III. Armed with these tools, the canonical transformations using Floquet theory could now be completely tackled.

2.2 Real Canonical Transformations

In working on this effort, it was found that the

standard transformations did not always yield real modal vectors, and an additional study in transforming complex eigensystems into real eigensystems was accomplished. This was a significantly easier task, in comparison to the first, as much has been written in linear algebra and matrix algebra textbooks on these types of transformations.

2.3 Perturbation Theory on Periodic Orbits

The only source for this part of the study was the work by Ross where he states "It is unique to use the eigenvectors and Poincaré exponents of the periodic trajectory, to canonically transform the generic equations of motion into nearly-periodic ones" (Ross, 1991:3). While Ross's work followed the theory of Dr. Wiesel, his was indeed the first to significantly analyze the possibilities of these methods.

III. Theory

3.1 Classical Floquet Theory

In classical Floquet theory, given a Hamiltonian system where $H(\bar{\mathbf{x}}, t)$ is the Hamiltonian function, and the state vector is

$$\bar{\mathbf{x}}^T = (q_1, p_1, q_2, p_2, \dots, q_n, p_n) , \quad i=1, n \quad (1)$$

The Hamiltonian equations of motion are then defined as

$$\dot{q}_i = \frac{\partial H}{\partial p_i} \quad \text{and} \quad \dot{p}_i = -\frac{\partial H}{\partial q_i} , \quad i=1, n \quad (2)$$

for each coordinate, q_i , and its conjugate momenta, p_i .

This can be more compactly written as

$$\dot{\bar{\mathbf{x}}} = \mathbf{Z} \frac{\partial H}{\partial \bar{\mathbf{x}}} \quad (3)$$

where $\mathbf{Z}$ is a correlation matrix defined as

$$\mathbf{Z} = \begin{bmatrix} 0 & 1 & 0 & 0 & \dots \\ -1 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 1 & \dots \\ 0 & 0 & -1 & 0 & \\ \vdots & \vdots & & & \ddots \end{bmatrix} \quad (4)$$

and $\mathbf{Z}$ is of order $2N$ (the number of coordinates and momenta in the state vector) (Wiesel, 1981:232,233). Since the

matrix $\mathbf{Z}$ is skew-symmetric and has a determinant equal to one, it follows the well established facts that $\mathbf{Z}^T = \mathbf{Z}^{-1} = -\mathbf{Z}$. The equations of variation then come from the linearization of the Hamiltonian equations of motion. This is accomplished by differentiating Eq(3) by the state vector yielding

$$\frac{\dot{\partial \bar{\mathbf{X}}}}{\partial \bar{\mathbf{X}}} = \mathbf{Z} \frac{\partial^2 H}{\partial \bar{\mathbf{X}}^2} = \mathbf{A}(t) \quad (5)$$

Here $\mathbf{A}(t)$ is the linearization of the dynamics and the equations of variation become

$$\dot{\bar{\mathbf{X}}} = \mathbf{A}(t) \bar{\mathbf{X}} = \mathbf{Z} \frac{\partial^2 H}{\partial \bar{\mathbf{X}}^2} \bar{\mathbf{X}} \quad (6)$$

(Ross, 1991:10,11). Here $\bar{\mathbf{x}}$ indicates coordinates on the *tangent space* of the system defined by Eq(3) (i.e., on a trajectory near the periodic motion). From this, a general solution, close to the periodic system of Eq(3), can be written using the fundamental matrix $\Phi(t, t_0)$, as

$$\bar{\mathbf{x}}(t) = \Phi(t, t_0) \bar{\mathbf{x}}(t_0) \quad (7)$$

where the fundamental matrix also obeys

$$\frac{d}{dt} (\Phi) = \mathbf{A}(t) \Phi = \mathbf{Z} \frac{\partial^2 H}{\partial \bar{\mathbf{X}}^2} \Phi \quad (8)$$

Given that the original system does include a periodic motion, then both Eqs(6) and (8) are time periodic linear differential equations. Note that the equations of variation actually arise from a first order Taylor series expansion of the original Hamiltonian about its nominal trajectory $\bar{\mathbf{x}}(t)$

$$\mathcal{H} = \frac{1}{2} \bar{\mathbf{x}}^T \frac{\partial^2 H}{\partial \bar{\mathbf{x}}^2} \bar{\mathbf{x}} \quad (9)$$

The main conclusion of classical Floquet theory then is that the periodic fundamental matrix Φ can be decomposed into

$$\Phi(t, t_0) = \mathbf{F}(t) e^{\mathbf{J}(t-t_0)} \mathbf{F}^{-1}(t_0) \quad (10)$$

where $\mathbf{F}(t)$ is periodic and $\mathbf{J}$ is a Jordan normal form.

The Floquet solution then is obtained by integration of Eqs(3) and (8) over one period of the motion. One result is the *monodromy matrix* $\Phi(\tau+t_0, t_0)$, where τ is the period. The matrix $\mathbf{F}(\tau)$ is then the system eigenvectors and $\exp\{\mathbf{J}\tau\}$ is the system eigenvalues at one period. More specifically, $\mathbf{J}$ is a Jordan normal matrix of Poincaré exponents, ω_i . These ω_i then describe the stability of the system (Pars, 1965:461-467). Once $\Phi(\tau+t_0, t_0)$ is found, and noting $\mathbf{F}(t_0+\tau) = \mathbf{F}(t_0)$ Eq(10) can be rewritten

$$\Phi = Fe^{J\tau}F^{-1} \quad (11)$$

where the time indices on Φ and F have been dropped due to their periodic nature. A standard math package can now be used on Φ to find the system eigenvalues, λ_i , and the system eigenvectors, $\mathbf{e}_i$. These eigenvectors are placed in the columns of the F matrix and the eigenvalues are the diagonal elements of the $\exp\{J\tau\}$ matrix. The diagonal elements of J are then found using

$$\omega_i = \frac{1}{\tau} \ln(\lambda_i) \quad (12)$$

or more conveniently for complex eigenvalues,

$$\begin{aligned} \text{Re}(\omega_i) &= \frac{1}{\tau} \ln[\text{Re}(\lambda_i)^2 + \text{Im}(\lambda_i)^2] \\ \text{Im}(\omega_i) &= \frac{1}{\tau} \tan^{-1} \left\{ \frac{\text{Im}(\lambda_i)}{\text{Re}(\lambda_i)} \right\} \end{aligned} \quad (13)$$

It is worth noting at this time that the Poincaré exponents occur only in positive/negative pairs for a canonical system. This leads to exactly four possible types of Hamiltonian coordinate/momenta pairs: 1) positive/negative real ω_i , 2) positive/negative imaginary ω_i , 3) a pair of zero ω_i , and 4) a 'box' or positive/negative pair of complex conjugate ω_i . Only the first three will be examined in detail in this study.

Completion of the Floquet solution at this point requires knowledge of $\mathbf{F}(t)$ over one period. Differentiating Eq(11) and substituting in Eq(8) and (10) yields

$$\dot{\mathbf{F}}(t) = \mathbf{Z} \frac{\partial^2 H}{\partial \mathbf{X}^2} \mathbf{F}(t) - \mathbf{F}(t) \mathbf{J} \quad (14)$$

A complete solution to the equations of variation, through Eq(7), can now be characterized over one period of the motion. Knowledge of $\mathbf{F}$ over one period also allows for a transformation to a set of periodic modal variables, $\bar{\mathbf{y}}(t)$ through the relationship

$$\bar{\mathbf{y}}(t) = \mathbf{F}^{-1} \mathbf{x}(t) \quad (15)$$

by the following derivation

$$\begin{aligned} \mathbf{x}(t) &= \Phi(t, t_0) \mathbf{x}(t_0) \\ \mathbf{x}(t) &= \mathbf{F}(t) e^{\mathbf{J}(t-t_0)} \mathbf{F}^{-1}(t_0) \mathbf{x}(t_0) \\ \mathbf{F}^{-1}(t) \mathbf{x}(t) &= e^{\mathbf{J}(t-t_0)} \mathbf{F}^{-1}(t_0) \mathbf{x}(t_0) \\ \bar{\mathbf{y}}(t) &= e^{\mathbf{J}(t-t_0)} \bar{\mathbf{y}}(t_0) \end{aligned} \quad (16)$$

To be useful for further study, this transformation needs to be canonical.

3.2 Canonical Floquet Theory

The most important criterion for a canonical transformation is that the new Hamiltonian must also follow Eq(3), that is

$$\dot{\bar{\mathbf{Y}}} = \mathbf{Z} \frac{\partial K}{\partial \bar{\mathbf{Y}}} \quad (17)$$

where K is the new system Hamiltonian and $\bar{\mathbf{Y}}$ the new system state vector. In order to ensure this, the original system eigenvector matrix, $\mathbf{F}$, must satisfy the relation

$$\mathbf{Z} = \mathbf{F}^T \mathbf{Z} \mathbf{F} \quad (18)$$

In other words, this means that the new Hamiltonian equations of motion also follow Eq(2) for the new state vector and the *same correlation matrix*. The formal proof defines the eigenvector matrix that satisfies Eq(18) as the symplectic eigenvector matrix (Siegel and Moser: 1971, 99-101). As in other work with canonical systems, this transpose of $\mathbf{F}$ is a standard transpose and not a Hermitian transpose. Equation (18) for a canonical transformation is, unfortunately, only applied to *constant eigenvector* systems in Siegel and Moser. Proof of its usefulness in a periodic system must be shown before this study can continue. Since $\mathbf{Z}$ is a constant matrix,

$$\dot{\mathbf{Z}} = \dot{\mathbf{F}}^T \mathbf{Z} \mathbf{F} + \mathbf{F}^T \mathbf{Z} \dot{\mathbf{F}} \quad (19)$$

must equal zero. Combining Eqs(5), (14), (19), using the identity relations for $\mathbf{Z}$, and assuming Eq(18) holds true, (19) reduces to

$$-\mathbf{J}^T \mathbf{Z} - \mathbf{Z} \mathbf{J} = 0 \quad (20)$$

Since $\mathbf{J}$ and $\mathbf{Z}$ are constant over the period of the motion, direct substitution will prove Eq(20) true for either the degenerate or non-degenerate case. Therefore, Eq(18) also holds true for a periodic system of eigenvectors (Wiesel and Pohlen, 1992:7).

If $\mathbf{F}$ forms a canonical transformation, what is the new Hamiltonian? Differentiating Eq(16) yields

$$\dot{\bar{\mathbf{y}}}(t) = \mathbf{J} e^{\mathbf{J}(t-t_0)} \bar{\mathbf{y}}(t_0) = \mathbf{J} \bar{\mathbf{y}}(t) \quad (21)$$

where, as in equation (6), $\mathbf{J}$ must be of the form $\mathbf{A}(t)$ or

$$\mathbf{J} = \mathbf{A}(t) = \mathbf{Z} \frac{\partial^2 K}{\partial \bar{\mathbf{Y}}^2} = \mathbf{Z} \mathbf{S} \quad (22)$$

With $\mathbf{S}$ defined as $\partial^2 K / \partial \bar{\mathbf{Y}}^2$, the new equations of variation come from the new variational Hamiltonian

$$\mathfrak{K} = \frac{1}{2} \bar{\mathbf{y}}^T \mathbf{S} \bar{\mathbf{y}} \quad (23)$$

(Wiesel and Pohlen, 1992:7)

3.2.1 Symplectic Normalization for Real Poincaré Exponents/Independent Eigenvectors

For a first look at symplectic normalization, consider the case where each Poincaré exponent generates a unique real eigenvector. The symplectic eigenvectors, $\bar{\boldsymbol{\rho}}_i$, can

differ from the original eigenvectors, $\bar{\mathbf{e}}_1$, by only a constant d_1 . In matrix notation this would be

$$\mathbf{E} = \mathbf{F}\mathbf{D} \quad (24)$$

with $\mathbf{D}$ a diagonal matrix of constant multipliers, and $\mathbf{E}$ the symplectic eigenvector matrix. Substituting $\mathbf{F}\mathbf{D}$ into Eq(18) for $\mathbf{F}$ yields, after rearrangement,

$$\mathbf{D}^{-1}\mathbf{Z}\mathbf{D}^{-1} = \mathbf{F}^T\mathbf{Z}\mathbf{F} \quad (25)$$

The form of the left hand side of Eq(25) is

$$\mathbf{D}^{-1}\mathbf{Z}\mathbf{D}^{-1} = \begin{bmatrix} 0 & \frac{1}{d_1 d_2} & 0 & 0 & \dots \\ -\frac{1}{d_1 d_2} & 0 & 0 & 0 & \dots \\ 0 & 0 & \ddots & & \\ 0 & 0 & & 0 & \frac{1}{d_{i-1} d_i} \\ \vdots & \vdots & & -\frac{1}{d_{i-1} d_i} & 0 \end{bmatrix}, \quad i=1, 2N \quad (26)$$

If the d_i are selected to satisfy Eq(25) then $\mathbf{D}$ will transform $\mathbf{F}$ into the symplectic $\mathbf{E}$. This then allows some freedom in the selection of the d_i . Siegel and Moser suggest the selection of one for all the odd d_i , which then uniquely determines the corresponding even d_i . This method generates one possible multiplier matrix, $\mathbf{D}$, and through Eq(24) a symplectic transformation matrix, $\mathbf{E}$, for any system

with unique real ω_i (Siegel and Moser, 1971:101).

3.2.2 Symplectic Normalization for Imaginary Poincaré Exponents/Complex Conjugate Eigenvectors

Unfortunately, not all of a system's ω_i pairs can be normalized as easily as pairs of positive/negative reals with real eigenvectors. In the case of imaginary pairs, where the associated eigenvectors will be complex conjugate pairs, the procedure as outlined by Siegel and Moser would result in two different multipliers that would destroy the complex conjugate nature of the eigenvectors. Fortunately, this problem is easy to solve. If we look at one particular multiplier pair entry in Eq(26)

$$(\mathbf{D}^{-1}\mathbf{ZD}^{-1})_{12} = \frac{1}{d_1 d_2} = (\mathbf{F}^T \mathbf{Z} \mathbf{F})_{12} \quad (27)$$

where the subscripts on $\mathbf{F}^T \mathbf{Z} \mathbf{F}$ and $\mathbf{D}^{-1}\mathbf{ZD}^{-1}$ signify the row and column respectively of these matrix products. For this to result in a symplectic transformation and in order to maintain the complex conjugate eigenvectors, d_1 must equal d_2 . Therefore the multiplier for both eigenvectors, $d_{1,2}$, is found simply from

$$d_{1,2} = \left(\frac{1}{(\mathbf{F}^T \mathbf{Z} \mathbf{F})_{12}} \right)^{\frac{1}{2}} \quad (28)$$

The fact that the scale factors are coupled is not surprising. While the eigenvectors of a linear system

can usually be normalized independently, in a canonical problem they must be normalized as canonical pairs. (Wiesel and Pohlen, 1992:10)

Therefore, this method is also appropriate for use on a pair of eigenvectors associated with real ω_1 .

3.2.3 *Symplectic Normalization for a Pair of Zero Poincaré Exponents/Repeated Eigenvectors*

This study would not be complete without a look at the degenerate case of a pair of zero ω_1 . The degenerate case commonly occurs in a Hamiltonian system with conserved quantities. Equally common is a rank deficiency in the eigenvector matrix when a pair of zeros is encountered. That is, there will be a repeated eigenvector (Wiesel and Pohlen, 1992:10).

3.2.3.1 *Determination of Generalized Eigenvector*

The first task in the symplectic normalization of the degenerate case is to determine the generalized or extended eigenvector that removes the rank deficiency in $\mathbf{F}$. The generalized eigenvector, $\mathbf{E}_{1,1}$, is found using

$$[\Phi - \lambda_1 \mathbf{I}] \mathbf{E}_{1,1} = \mathbf{E}_1 \quad (29)$$

for a constant coefficient case (Reid, 1983:346). But in a periodic system where the form of $\exp(\mathbf{J}\tau)$ and the eigenvalue matrix are, respectively,

$$\mathbf{J} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad e^{\mathbf{J}\tau} = \Lambda = \begin{bmatrix} 1 & \tau \\ 0 & 1 \end{bmatrix} \quad (30)$$

and the more appropriate form of Eq(29) is then written

$$[\Phi - \lambda_i \mathbf{I}] \mathbf{E}_{i+1} = \tau \mathbf{E}_i \quad (31)$$

As a check on the new $\mathbf{F}$ with the extended eigenvector, the system should satisfy $\Phi \mathbf{F} - \mathbf{F} \Lambda = \mathbf{0}$. Note from Eq(31) that the generalized eigenvector is not entirely arbitrary and therefore cannot be normalized in the standard fashion. It will, however, be indeterminable to an additive multiple of the repeated eigenvector. In fact,

$$\mathbf{E}'_{i+1} = \mathbf{E}_{i+1} + \alpha \mathbf{E}_i \quad (32)$$

defines a generalized eigenvector for any value of α (Wiesel and Pohlen, 1992:10).

3.2.3.2 Normalization for the Degenerate Case

Because of the arbitrary value of α , the multiplier matrix, $\mathbf{D}$, is not necessarily diagonal. Instead it will be of the form

$$\mathbf{D} = \begin{bmatrix} d_1 & \alpha \\ 0 & d_2 \end{bmatrix} \quad (33)$$

for the repeated/generalized eigenvector pair. Since $\mathbf{D}$ is no longer diagonal, Eq(25) will also change to

$$(\mathbf{D}^T)^{-1} \mathbf{Z} \mathbf{D}^{-1} = \mathbf{F}^T \mathbf{Z} \mathbf{F} \quad (34)$$

since $\mathbf{D}^*$ is no longer equal to $\mathbf{D}$. After the required math on the left hand side of Eq(34), the result for a degenerate pair of eigenvectors is the same as that for any other pair in Eq(26), or

$$(\mathbf{D}^T)^{-1} \mathbf{Z} \mathbf{D}^{-1} = \begin{bmatrix} 0 & \frac{1}{d_1 d_2} \\ -\frac{1}{d_1 d_2} & 0 \end{bmatrix} \quad (35)$$

Equation (35) demonstrates that the arbitrary multiple α really is arbitrary, and is just as well chosen as zero. In the degenerate case, the multipliers d_1 and d_2 must be the same, due to the specific relationship between the repeated eigenvector and the generalized eigenvector defined in Eq(31). Whereas in the case of a pair of complex conjugate eigenvectors, the multipliers are chosen to be the same for the convenience of maintaining conjugate pairs of eigenvectors.

Therefore, the symplectic normalization for each pair of eigenvectors follows the same procedure. First the matrix of values $\mathbf{F}^* \mathbf{Z} \mathbf{F}$ are found. Then multipliers are found by applying Eq(28) to each pair of eigenvectors.

The value for a $(\mathbf{F}^* \mathbf{Z} \mathbf{F})_{i,i+1}$ pair will commonly be a negative or a complex number. This inevitably results in a set of symplectic eigenvectors that are complex and thus a set of complex modal variables. This becomes very inconvenient for analysis of the modal variables.

3.3 Real Valued Symplectic Eigenvectors

Looking at the new variational Hamiltonian as defined in Eq(23), the matrix **S** can be found with the aid of Eq(22), since

$$\mathbf{S} = \mathbf{Z}^{-1}\mathbf{J} = -\mathbf{Z}\mathbf{J} \quad (36)$$

where again, **S** is defined as $\partial^2 K / \partial \tilde{\mathbf{Y}}^2$, or the second partial of the new Hamiltonian with respect to the new state vector. It will be the variational Hamiltonian, **K**, rather than the system Hamiltonian, **K**, that will be of concern in the following sections. Therefore, the value of **S** will be of primary importance.

3.3.1 Real Transformations for Non-degenerate Cases

For any system without a degenerate mode, the **J** matrix will be of the form

$$\mathbf{J} = \begin{bmatrix} \omega_i & 0 \\ 0 & -\omega_i \end{bmatrix} \quad (37)$$

Using Eq(37) in Eq(36), **S** is of the form

$$\mathbf{S}_i = \begin{bmatrix} 0 & -\omega_i \\ \omega_i & 0 \end{bmatrix} \quad (38)$$

The variational Hamiltonian of Eq(23) can then be written as

$$\mathcal{R} = \frac{1}{2} \mathbf{y}^T \mathbf{S} \mathbf{y} = \sum_{i=1}^{2N-1} \omega_i y_i y_{i+1}, \quad i=\text{odd integers} \quad (39)$$

for each pair of Poincaré exponents. Equation (39) provides the following equations of variation

$$\begin{aligned} \dot{y}_i &= \frac{\partial \mathcal{R}}{\partial y_{i+1}} = \omega_i y_i \quad (i=\text{odd integers}) \\ \dot{y}_{i+1} &= -\frac{\partial \mathcal{R}}{\partial y_i} = -\omega_i y_{i+1} \quad (i=\text{odd integers}) \end{aligned} \quad (40)$$

which have the simple solutions

$$y_i = y_{i_0} e^{\omega_i t} \quad y_{i+1} = y_{i+1_0} e^{-\omega_i t} \quad (41)$$

Again, this is easily applied when ω_i are real, but for imaginary ω_i the result is a pair of complex valued modal variables. This results in the need for another transformation to ensure real modal variables. A transformation will also be required to produce real symplectic eigenvectors in order to make integrations of the eigenvectors easier to implement in computer code. In some cases, these transformations can be accomplished with the same transformation matrix.

The matrix that will transform the imaginary pairs of ω_i in the $\mathbf{J}$ matrix is defined as

$$\mathbf{T} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \quad (42)$$

(Strang, 1988:298). Recall that any transformation to a canonical system must obey Eq(18); therefore in this transformation, $\mathbf{T}^T \mathbf{Z} \mathbf{T}$ must equal $\mathbf{Z}$. To meet this requirement, Eq(42) is found to be

$$\mathbf{T}_{A_i} = \frac{1}{\sqrt{2\omega_i}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \quad (43)$$

and will be defined as a type A transformation matrix. Using $\mathbf{T}_{A_i}$ in Eq(11) becomes

$$\Phi = \mathbf{E} \mathbf{T}_{A_i}^{-1} e^{(\mathbf{T}_{A_i} \mathbf{J} \mathbf{T}_{A_i}^T) \tau} \mathbf{T}_{A_i} \mathbf{E}^{-1} \quad (44)$$

The matrices $\mathbf{E}$, $\mathbf{J}$, and $\mathbf{S}$ are also redefined as

$$\mathbf{E}' = \mathbf{E} (\mathbf{T}_{A_i})^{-1} \quad (45)$$

$$\mathbf{J}'_i = \mathbf{T}_{A_i} \mathbf{J}_i (\mathbf{T}_{A_i})^{-1} = \begin{bmatrix} 0 & \omega_i \\ -\omega_i & 0 \end{bmatrix} \quad (46)$$

$$\mathbf{S}'_i = -\mathbf{Z} \mathbf{J}'_i = \begin{bmatrix} \omega_i & 0 \\ 0 & \omega_i \end{bmatrix} \quad (47)$$

Although this transformation creates a real $\mathbf{J}$ and therefore a real $\mathbf{S}$, the resulting symplectic eigenvectors will become either purely real or purely imaginary. Ensuring that these eigenvectors are real can be accomplished through another transformation matrix, or a type B transformation defined as

$$\mathbf{T}_{B_i} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad (48)$$

Following the same method as with a type A transformation, $\mathbf{E}$, $\mathbf{J}$, and $\mathbf{S}$ become

$$\mathbf{E}'' = \mathbf{E}' (\mathbf{T}_{B_i})^{-1} \quad (49)$$

$$\mathbf{J}_i'' = \mathbf{T}_{B_i} \mathbf{J}_i' (\mathbf{T}_{B_i})^{-1} = \begin{bmatrix} 0 & -1\omega \\ 1\omega & 0 \end{bmatrix} \quad (50)$$

$$\mathbf{S}_i'' = -\mathbf{Z} \mathbf{J}_i'' = \begin{bmatrix} -1\omega & 0 \\ 0 & -1\omega \end{bmatrix} \quad (51)$$

A fully populated $\mathbf{T}_A$ matrix will be block diagonal with a $\mathbf{T}_{A_i}$ for each pair of imaginary ω_i and an identity matrix for all other pairs of ω_i . Likewise, a fully populated $\mathbf{T}_B$ matrix will be block diagonal with a $\mathbf{T}_{B_i}$ for each pair of imaginary $\bar{\rho}_i$ and an identity matrix for all other pairs of $\bar{\rho}_i$.

3.3.3 Real Transformations for Degenerate Cases

In the degenerate case, the $\mathbf{J}_1$ submatrix will always be real, so a type A transformation will never be necessary. As in the non-degenerate case, the symplectic $\tilde{\mathbf{p}}_1$ can be purely imaginary as a result of symplectic normalization, and therefore a type B transformation should be applied. The effect on $\mathbf{E}$ will be the same as that in Eq(49), but since the original form of $\mathbf{J}$ and $\mathbf{S}$ in the degenerate case are

$$\mathbf{J}_1 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \text{ and } \mathbf{S}_1 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \quad (52)$$

the transformation results in

$$\mathbf{J}_1'' = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \text{ and } \mathbf{S}_1'' = \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix} \quad (53)$$

3.4 Modal Variables and $\mathbf{E}$ After Real Canonical Transformation

While Eq(23) is still valid for the variational Hamiltonian, the specific form for the i th portion of Eq(38) will now have six different possibilities. These six different variations are outlined in Appendix A.

The final results of the last several sections is that Floquet theory can be applied to any time periodic system and result in a real canonical transformation to modal variables through $\tilde{\mathbf{x}}(t) = \mathbf{E}(t)\tilde{\mathbf{y}}(t)$. There will also be a

Jordan-like normal form $\mathbf{J}$ that is real valued over the period of the system (Wiesel and Pohlen, 1992:14). Regardless of the specific form for the variational equations of motion, the time periodic symplectic eigenvectors can be characterized over one period through Eq(14) and will remain real over that period.

In characterizing the motion of the system, the initial state vector, $\bar{\mathbf{x}}$, and the transformation matrix, $\mathbf{E}$, can be integrated over one period with their values sampled at regular intervals. These values are then converted into a set of Fourier coefficients to be used in construction of the perturbation solutions (Brower and Clemence, 1961:108-112). "The advantage of the Fourier representation is that the coefficients may be reassembled into the periodic orbit at any time necessary" (Ross, 1991:28).

3.5 *The Restricted Three-Body Problem*

With the theory for canonical Floquet theory developed, it becomes important to examine a specific periodic system to validate the usefulness of the theory. In the case of this study, the restricted three-body problem presented by Ross will be used. The particular system examined is the Sun-Jupiter system. Only the pertinent equations and information will be presented here, while the reader is referred to Ross (1991) for detailed derivations.

The definition of the restricted three-body problem was first presented by Euler in 1772. The problem is defined

as:

Two bodies revolve around their center of mass in circular orbits under the influence of the mutual gravitational attraction and a third body (attracted by the previous two but not influencing their motion) moves in the plane defined by the two revolving bodies. *The restricted problem of three bodies is to describe the motion of this third body.* (Szebehely, 1967:8)

In setting up the three-body problem, Ross first defines several non-dimensional quantities. These are:

$$m_1 = \frac{M_1}{M_1+M_2} , \quad m_2 = \frac{M_2}{M_1+M_2} , \quad m_3 = \frac{M_3}{M_1+M_2} \approx 0 \quad (54)$$

$$s_1 = \frac{S_1}{S_1+S_2} , \quad s_2 = \frac{S_2}{S_1+S_2} \quad (55)$$

where M_1 and M_2 are the masses of the primary bodies and M_3 is the mass of the third body which is negligible. S_1 and S_2 are the distances from the primary bodies to the system center of mass. Since $s_1+s_2 = m_1+m_2 = 1$, these quantities are then redefined with a single dimensionless variable μ .

$$s_1 = m_2 = \mu , \quad s_2 = m_1 = 1-\mu \quad (56)$$

The relationship of these dimensionless quantities are shown in Figure 1 (Ross, 1991:4-6).

The Hamiltonian for the restricted three-body system is then defined by Ross as

$$H = \frac{1}{2} (p_1^2 + p_2^2) + p_1 q_2 - p_2 q_1 - \frac{1-\mu}{r_1} - \frac{\mu}{r_2} \quad (57)$$

where

$$\begin{aligned} r_1 &= [(q_1 - \mu)^2 + q_2^2]^{\frac{1}{2}} \\ r_2 &= [(q_1 + 1 - \mu)^2 + q_2^2]^{\frac{1}{2}} \end{aligned} \quad (58)$$

and the system state vector is

$$\mathbf{x}^T(t) = [q_1(t), p_1(t), q_2(t), p_2(t)] \quad (59)$$

The resulting system equations of motion are

$$\begin{aligned} \dot{q}_1 &= \frac{\partial H}{\partial p_1} = p_1 + q_2 \\ \dot{p}_1 &= -\frac{\partial H}{\partial q_1} = p_2 - \frac{(1-\mu)(q_1 - \mu)}{r_1^3} - \frac{\mu(q_1 + 1 - \mu)}{r_2^3} \\ \dot{q}_2 &= \frac{\partial H}{\partial p_2} = p_2 - q_1 \\ \dot{p}_2 &= -\frac{\partial H}{\partial q_2} = -p_1 - \frac{(1-\mu)q_2}{r_1^3} - \frac{\mu q_2}{r_2^3} \end{aligned} \quad (60)$$

3.5.1 Periodic Orbits and the Equations of Variation

The definition for a periodic orbit is simply stated as the state vector at time τ is equal to the state vector at the initial time, or

$$\mathbf{x}(0) = \mathbf{x}(\tau) \quad (61)$$

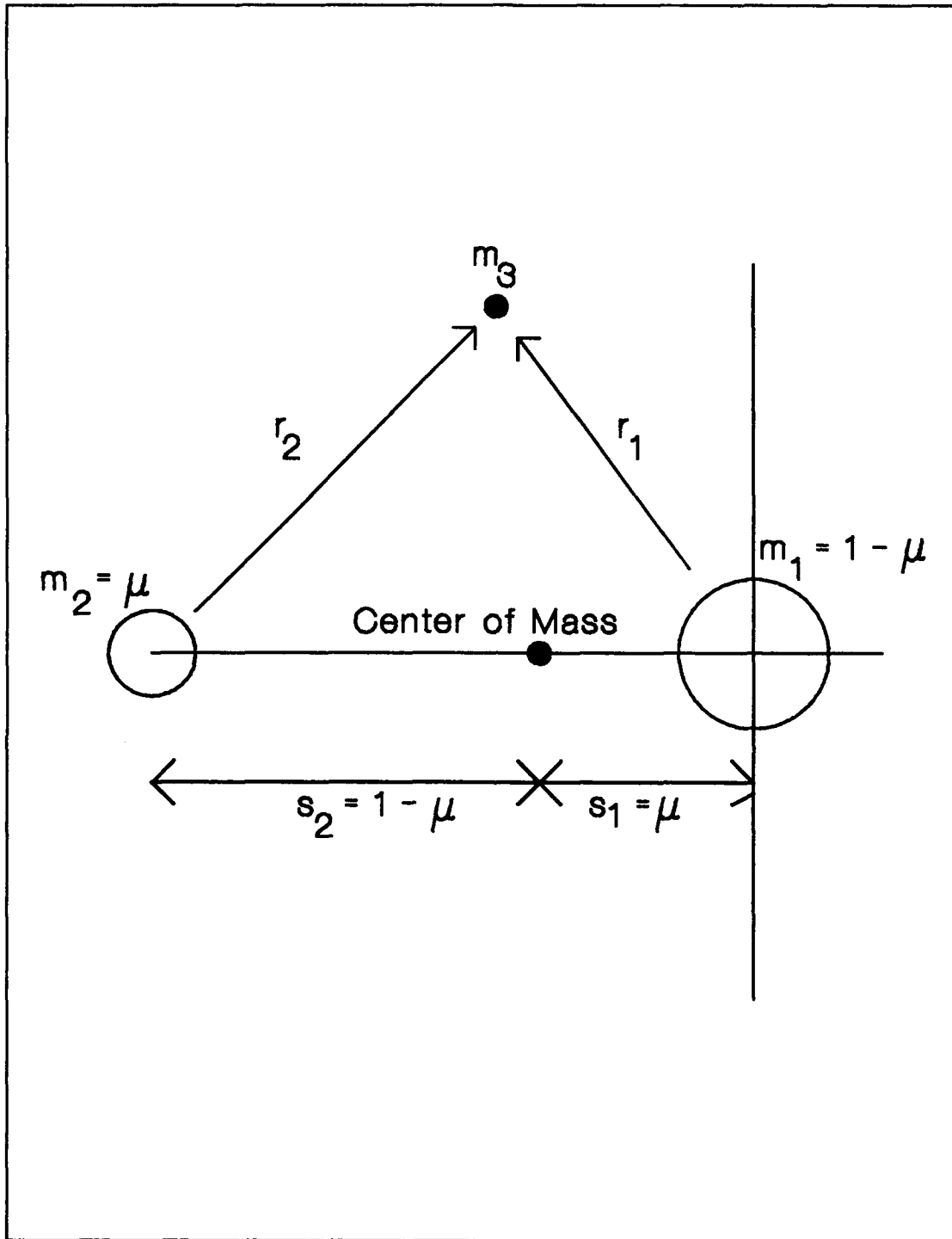


Figure 1: Reference Frame for the Restricted Three-Body System (Ross, 1991:6)

Using the periodicity of a system, a solution for the initial condition can be found by "iteratively narrowing the difference between the initial and final conditions" (Ross, 1991:9). In order to make the appropriate adjustments to each iteration, the behavior of nearly periodic orbits must be known. The equations of variation describe the motion of these nearby trajectories. As defined by Eqs(4), (5), and (6), the equations of variation for the three-body problem are given as

$$\dot{\mathbf{x}} = \begin{bmatrix} \delta \dot{q}_1 \\ \delta \dot{p}_1 \\ \delta \dot{q}_2 \\ \delta \dot{p}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ -H_{11} & 0 & -H_{13} & 1 \\ -1 & 0 & 0 & 1 \\ -H_{31} & -1 & -H_{33} & 0 \end{bmatrix} \begin{bmatrix} \delta q_1 \\ \delta p_1 \\ \delta q_2 \\ \delta p_2 \end{bmatrix} = \mathbf{A}(t) \mathbf{x} \quad (62)$$

where

$$\begin{aligned} H_{11} &= \frac{-3(q_1 - \mu)^2(1 - \mu)}{r_1^5} - \frac{3(q_1 + 1 - \mu)^2\mu}{r_2^5} + \frac{1 - \mu}{r_1^3} + \frac{\mu}{r_2^3} \\ H_{13} &= \frac{-3(q_1 - \mu)q_2(1 - \mu)}{r_1^5} - \frac{3(q_1 + 1 - \mu)q_2\mu}{r_2^5} \\ H_{31} &= H_{13} \\ H_{33} &= \frac{-3q_2^2(1 - \mu)}{r_1^5} - \frac{3q_2^2\mu}{r_2^5} + \frac{1 - \mu}{r_1^3} + \frac{\mu}{r_2^3} \end{aligned} \quad (63)$$

These variational equations of motion form a set of time-varying, linear, differential equations which then follow Eqs(7) and (8) in forming a fundamental set of solutions, Φ . Equations (60) and (62) can be numerically integrated to form a solution to Eq(7) in the form of

$$\begin{bmatrix} \delta q_1(\tau) \\ \delta p_1(\tau) \\ \delta q_2(\tau) \\ \delta p_2(\tau) \end{bmatrix} = \begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} & \phi_{14} \\ \phi_{21} & \phi_{22} & \phi_{23} & \phi_{24} \\ \phi_{31} & \phi_{32} & \phi_{33} & \phi_{34} \\ \phi_{41} & \phi_{42} & \phi_{43} & \phi_{44} \end{bmatrix} \begin{bmatrix} \delta q_1(0) \\ \delta p_1(0) \\ \delta q_2(0) \\ \delta p_2(0) \end{bmatrix} \quad (64)$$

3.5.2 Determination of Initial Conditions

In finding a periodic orbit, any set of initial conditions can be inserted into Eq(59) and integrated for a desired τ . The error in the final state vector is the left hand side of Eq(64) and after inverting the integrated Φ , the error in the initial state vector is found.

Since the Hamiltonian state variables are interdependent, this method of iteration is overly cumbersome in solving for the initial conditions. The state variables, Eq(59), can be shown to be functions of H , μ , q_1 , and q_2 only (Ross, 1991:17-18). It makes sense to maintain the conserved quantity, H , constant since it will allow for better comparisons in the perturbation study. The quantity μ is system dependent and therefore constant. This leaves us with only the coordinates q_1 and q_2 to manipulate. After some examination it is noted that selection of $q_2=0$ also yields $p_1=0$, or in other words, the orbit starting point will be on the q_1 axis in a motion perpendicular to the q_1 axis. The result of this selection for initial conditions is that Eq(64) can be reduced to

$$\begin{bmatrix} \delta p_1(\tau) \\ \delta q_2(\tau) \end{bmatrix} = \begin{bmatrix} \phi_{21} & \phi_{24} \\ \phi_{31} & \phi_{34} \end{bmatrix} \begin{bmatrix} \delta q_1(0) \\ \delta p_2(0) \end{bmatrix} \quad (65)$$

(Ross, 1991:13).

But since the selection of p_2 is not arbitrary, another revision, as outlined in discussions with Dr. Wiesel, replaces p_2 with another selectable initial condition, the integration time. This selection then yields another quantity, the period, that will be as important as the initial coordinates and momenta. Equation (65) can now be modified to

$$\begin{bmatrix} \delta p_1(\tau) \\ \delta q_2(\tau) \end{bmatrix} = \begin{bmatrix} \frac{\partial p_1(\tau)}{\partial q_1(0)} & \frac{\partial p_1(\tau)}{\partial \tau} \\ \frac{\partial q_2(\tau)}{\partial q_1(0)} & \frac{\partial q_2(\tau)}{\partial \tau} \end{bmatrix} \begin{bmatrix} \delta q_1(0) \\ \delta \tau \end{bmatrix} \quad (66)$$

where

$$\begin{aligned} \frac{\partial p_1(\tau)}{\partial q_1(0)} &= \left. \frac{\partial p_1(\tau)}{\partial q_1(0)} \right|_{direct} + \left. \frac{\partial p_1(\tau)}{\partial p_2(0)} \right|_{\tau} \left. \frac{\partial p_2(0)}{\partial q_1(0)} \right|_0 = \phi_{21} + \phi_{24} \frac{\dot{p}_1}{\dot{q}_2} \\ \frac{\partial p_1(\tau)}{\partial \tau} &= \dot{p}_1 \end{aligned} \quad (67)$$

and likewise

$$\frac{\partial q_2(\tau)}{\partial q_1(0)} = \phi_{31} + \phi_{34} \frac{\dot{p}_1}{\dot{q}_2} \quad (68)$$

$$\frac{\partial q_2(\tau)}{\partial \tau} = \dot{q}_2$$

The non-direct term in the first equations of Eq(68) and (69) are due to the coupling effect of q_1 and p_2 . Through inversion of Eq(67) the required changes in q_1 and the period can now be found.

3.5.3 Surface of Section

Finding an initial guess for q_1 and τ is made easier through the use of surface of section plots. These plots define regions of periodic motion by plotting only the apogee and perigee points along a periodic trajectory. A collection of these plots was compiled into Jefferys' *An Atlas of Surface of Section for the Restricted Problem of Three Bodies*. As an example of a surface of section plot, Figure 2 shows a precessing elliptical trajectory. Given enough time, a plot of only the perigee and apogee points for an infinite set of ellipses will form a set of rings as seen in figure 3. Not all periodic regions will form a pair of rings, but it is the easiest form to understand. To examine other periodic regions the reader is directed to Jefferys (1971). These regions can then be searched for specific periodic trajectories.

Since the development of the equations of motion in

Jefferys differed from that of Ross (as adapted from Szebehely), Ross describes the transformation between the two derivations (Ross, 1991:14-17). The purpose of this transformation was so that the initial conditions used to create Jefferys atlas could be used as input for Szebehely's equations of motion. The resulting surface of sections could be directly compared to the atlas developed by Jefferys.

For the Sun-Jupiter system, $\mu=0.00095388$. With the selection of 3.15 for the Jacobian constant, J , the resulting surface of section can be seen in figure 4. The relationship between the Jacobian constant and the system Hamiltonian is

$$H = \frac{1}{2} [\mu(1-\mu) - J] \quad (69)$$

(Ross, 1991:16).

3.6 Perturbation Theory on the Restricted Three-Body Problem

With the assistance of the surface of section plot and the iterative integration of the equations of motion, a periodic trajectory is found. The next step is to find out what effect perturbations to the initial conditions have on the stability of the orbit.

In examination of the effect of perturbations to the initial conditions, two different representations of the

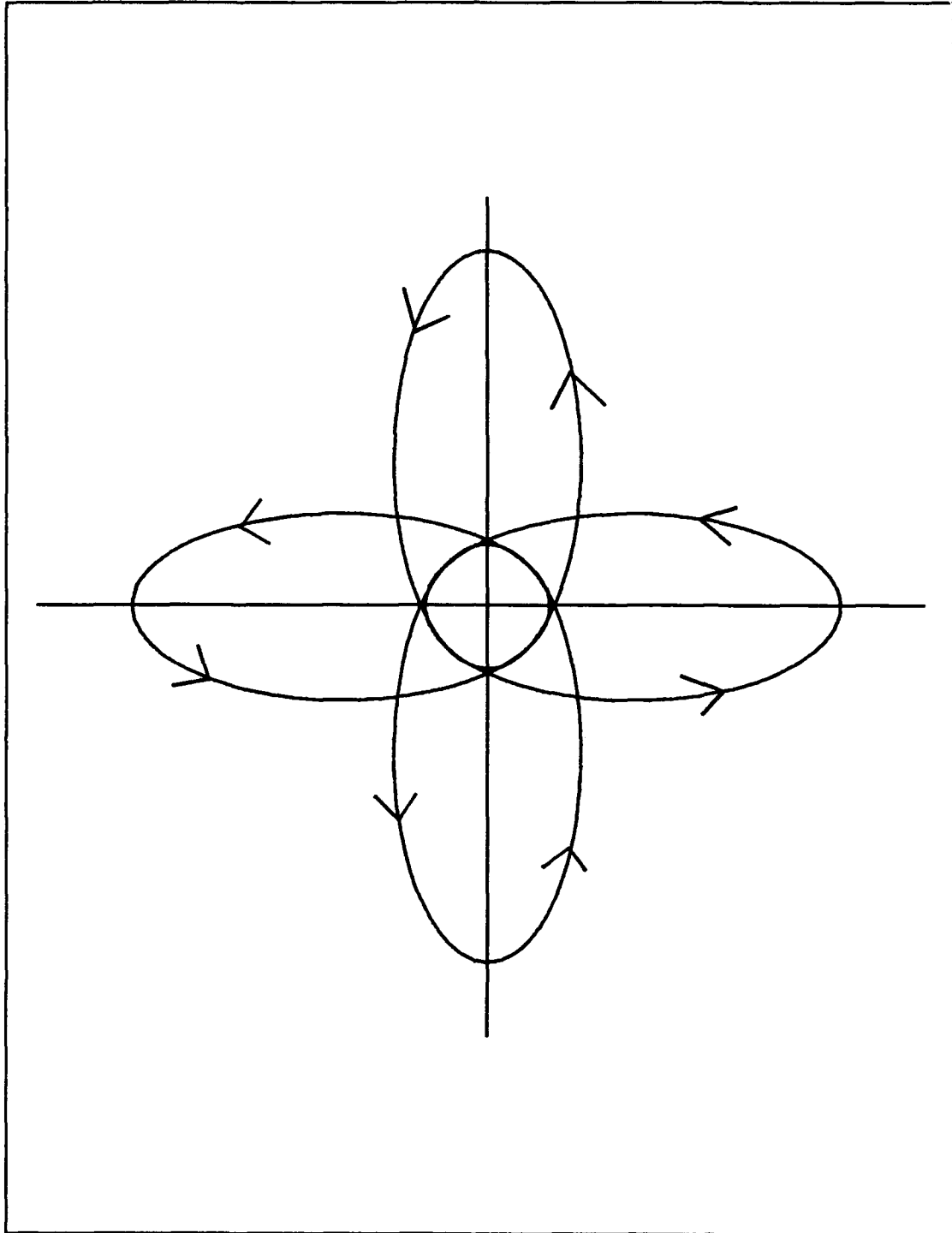


Figure 2: An Elliptical Trajectory Precessing
About Primary $1-\mu$ (Ross, 1991:19)

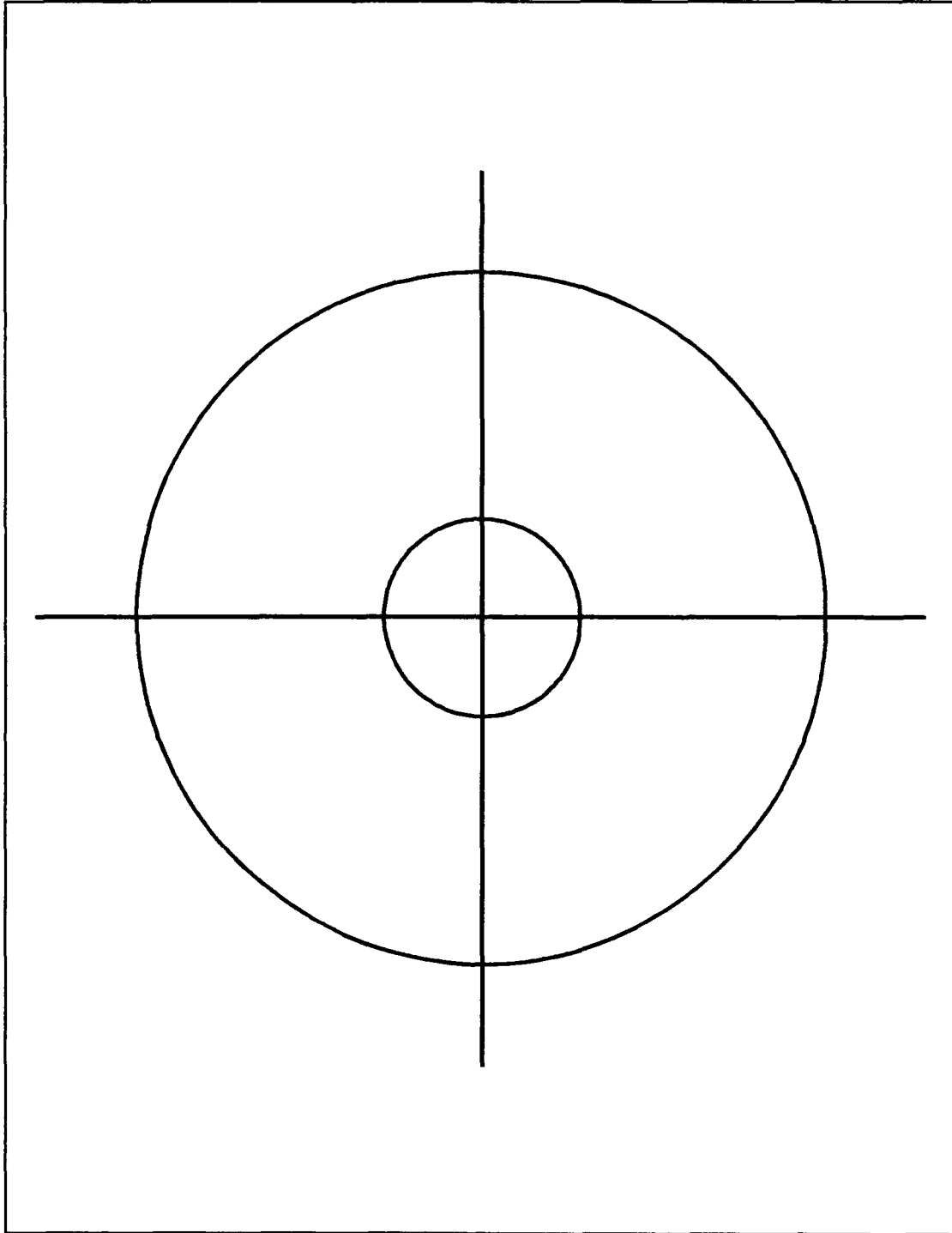


Figure 3: The Surface of Section of the
Elliptical Trajectory (Ross, 1991:20)

$J = 3.15$

$\mu = 0.00095388$

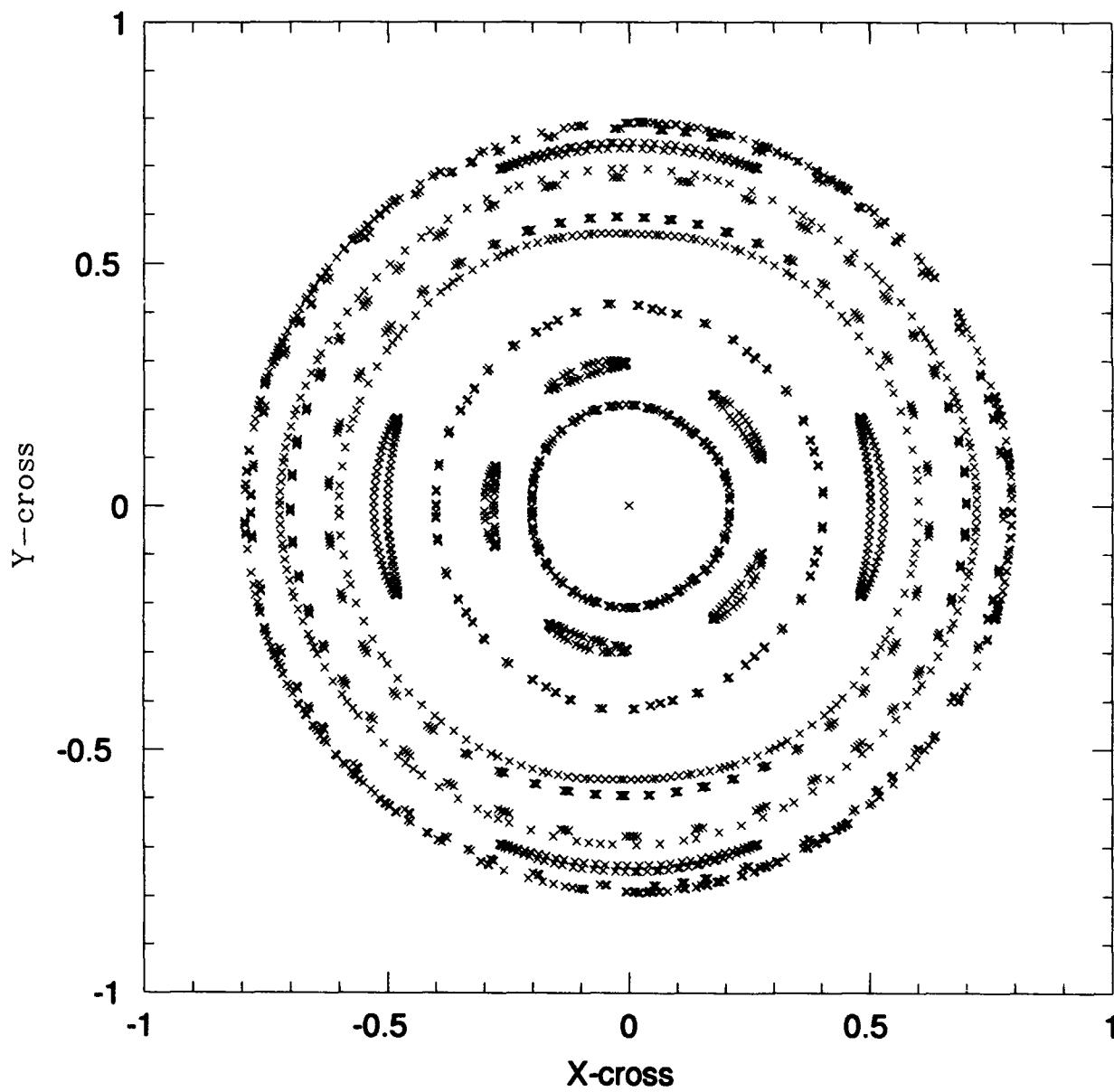


Figure 4: The Surface of Section for the Sun-Jupiter System

nearly periodic orbit will be constructed. Both representations will use the Φ found from the integration of the periodic initial conditions for one period. The Φ matrix will be transformed into a real valued, symplectic eigenvector matrix, $\mathbf{E}$, and a Jordan normal-like matrix, $\mathbf{J}$, of Poincaré exponents.

3.6.1 The Exact Representation

In the first representation, the periodic $\tilde{\mathbf{x}}$ and $\mathbf{E}$ will be integrated over one period and an evenly spaced sampling of these quantities will be converted into a set of one hundred Fourier coefficients. A nearly periodic trajectory, close to the original trajectory, will then be integrated over time. The integration time required for the nearly periodic trajectory will be many times the orbital period of the system and is on the order of the period of an oscillatory Poincaré exponent. Subtracting the periodic trajectory from the nearly periodic defines an $\tilde{\mathbf{x}}(t)$ (as described in Eq(6)), and using the canonical Floquet transformation,

$$\mathbf{y}(t) = \mathbf{E}^{-1}(t)\tilde{\mathbf{x}}(t) \quad (70)$$

creates a set of modal variables over the integration time. This set of modal variables can then be plotted as an exact representation of the nearly periodic orbit (Ross, 1991:24).

3.6.2 The Expanded Representation

For the second representation, the original Hamiltonian

is canonically transformed into the modal variables and expanded in a Taylor series which produces

$$K(\mathbf{y}) = H(\mathbf{0}) + \sum_{i=1}^4 \frac{\partial H(\mathbf{y})}{\partial y_i} \Big|_{\mathbf{y}=\mathbf{0}} y_i + \frac{1}{2!} \sum_{j=1}^4 \sum_{i=1}^4 \frac{\partial^2 H(\mathbf{y})}{\partial y_i \partial y_j} \Big|_{\mathbf{y}=\mathbf{0}} y_i y_j + \dots \quad (71)$$

where $\mathbf{y}=\mathbf{0}$ centers the expansion on the periodic trajectory.

In tensor notation Eq(71) becomes

$$K(\mathbf{y}) = H(\mathbf{0}) + H_1(\mathbf{0}) y_1 + \frac{1}{2!} H_{1j}(\mathbf{0}) y_1 y_j + \frac{1}{3!} H_{1jk}(\mathbf{0}) y_1 y_j y_k + \dots \quad (72)$$

The first term in the expansion is the Hamiltonian for a periodic orbit, and is a constant. The second, or linear term is identically zero, because it describes the motion of the periodic trajectory with respect to itself. The third, or quadratic term is the Floquet problem, and becomes a constant coefficient, linear system in the new variables. Since the magnitude of the modal state vector is small compared to one, the expansion is truncated after the fourth term. (Ross, 1991:31-32)

The new modal Hamiltonian then becomes

$$K(\mathbf{y}) = H(\mathbf{0}) + \frac{1}{2} \mathbf{y}^T \mathbf{S} \mathbf{y} + \frac{1}{6} H_{1jk}(\mathbf{0}) y_1 y_j y_k \quad (73)$$

where the i th portion of the quadratic term will follow one of the forms outlined in Appendix A.

The third order tensor can be expanded to the very cumbersome form of

$$\begin{aligned}
\frac{1}{6} H_{ijk} Y_i Y_j Y_k = & c_1 Y_1^3 + c_2 Y_1^2 Y_2 + c_3 Y_1^2 Y_3 + c_4 Y_1^2 Y_4 + c_5 Y_1 Y_2^2 \\
& + c_6 Y_1 Y_2 Y_3 + c_7 Y_1 Y_2 Y_4 + c_8 Y_1 Y_3^2 + c_9 Y_1 Y_3 Y_4 + c_{10} Y_1 Y_4^2 \\
& + c_{11} Y_2^3 + c_{12} Y_2^2 Y_3 + c_{13} Y_2^2 Y_4 + c_{14} Y_2 Y_3^2 + c_{15} Y_2 Y_3 Y_4 \\
& + c_{16} Y_2 Y_4^2 + c_{17} Y_3^3 + c_{18} Y_3^2 Y_4 + c_{19} Y_3 Y_4^2 + c_{20} Y_4^3
\end{aligned} \tag{74}$$

where the coefficients c_i are defined by the triple summation $i=1$ to 4, $j=1$ to 4, and $k=1$ to 4 over the right hand sides of

$$\begin{aligned}
c_1(t) &= \frac{1}{6} H_{1jk} \rho_{11} \rho_{j1} \rho_{k1} \\
c_2(t) &= \frac{1}{6} H_{1jk} [\rho_{11} \rho_{j1} \rho_{k2} + \rho_{11} \rho_{j2} \rho_{k1} + \rho_{12} \rho_{j1} \rho_{k1}] \\
c_3(t) &= \frac{1}{6} H_{1jk} [\rho_{11} \rho_{j1} \rho_{k3} + \rho_{11} \rho_{j3} \rho_{k1} + \rho_{13} \rho_{j1} \rho_{k1}] \\
c_4(t) &= \frac{1}{6} H_{1jk} [\rho_{11} \rho_{j1} \rho_{k4} + \rho_{11} \rho_{j4} \rho_{k1} + \rho_{14} \rho_{j1} \rho_{k1}] \\
c_5(t) &= \frac{1}{6} H_{1jk} [\rho_{11} \rho_{j2} \rho_{k2} + \rho_{12} \rho_{j1} \rho_{k2} + \rho_{12} \rho_{j2} \rho_{k1}] \\
c_6(t) &= \frac{1}{6} H_{1jk} [\rho_{11} \rho_{j2} \rho_{k3} + \rho_{11} \rho_{j3} \rho_{k2} + \rho_{12} \rho_{j1} \rho_{k3} \\
&\quad + \rho_{12} \rho_{j3} \rho_{k1} + \rho_{13} \rho_{j1} \rho_{k2} + \rho_{13} \rho_{j2} \rho_{k1}] \\
c_7(t) &= \frac{1}{6} H_{1jk} [\rho_{11} \rho_{j2} \rho_{k4} + \rho_{11} \rho_{j4} \rho_{k2} + \rho_{12} \rho_{j1} \rho_{k4} \\
&\quad + \rho_{12} \rho_{j4} \rho_{k1} + \rho_{14} \rho_{j1} \rho_{k2} + \rho_{14} \rho_{j2} \rho_{k1}] \\
c_8(t) &= \frac{1}{6} H_{1jk} [\rho_{11} \rho_{j3} \rho_{k3} + \rho_{13} \rho_{j1} \rho_{k3} + \rho_{13} \rho_{j3} \rho_{k1}] \\
c_9(t) &= \frac{1}{6} H_{1jk} [\rho_{11} \rho_{j3} \rho_{k4} + \rho_{11} \rho_{j4} \rho_{k3} + \rho_{13} \rho_{j1} \rho_{k4} \\
&\quad + \rho_{13} \rho_{j4} \rho_{k1} + \rho_{14} \rho_{j1} \rho_{k3} + \rho_{14} \rho_{j3} \rho_{k1}] \\
c_{10}(t) &= \frac{1}{6} H_{1jk} [\rho_{11} \rho_{j4} \rho_{k4} + \rho_{14} \rho_{j1} \rho_{k4} + \rho_{14} \rho_{j4} \rho_{k1}] \\
c_{11}(t) &= \frac{1}{6} H_{1jk} \rho_{12} \rho_{j2} \rho_{k2} \\
c_{12}(t) &= \frac{1}{6} H_{1jk} [\rho_{12} \rho_{j2} \rho_{k3} + \rho_{12} \rho_{j3} \rho_{k2} + \rho_{13} \rho_{j2} \rho_{k2}] \\
c_{13}(t) &= \frac{1}{6} H_{1jk} [\rho_{12} \rho_{j2} \rho_{k4} + \rho_{12} \rho_{j4} \rho_{k2} + \rho_{14} \rho_{j2} \rho_{k2}] \\
c_{14}(t) &= \frac{1}{6} H_{1jk} [\rho_{12} \rho_{j3} \rho_{k3} + \rho_{13} \rho_{j2} \rho_{k3} + \rho_{13} \rho_{j3} \rho_{k2}] \\
c_{15}(t) &= \frac{1}{6} H_{1jk} [\rho_{12} \rho_{j3} \rho_{k4} + \rho_{12} \rho_{j4} \rho_{k3} + \rho_{13} \rho_{j2} \rho_{k4} \\
&\quad + \rho_{13} \rho_{j4} \rho_{k2} + \rho_{14} \rho_{j2} \rho_{k3} + \rho_{14} \rho_{j3} \rho_{k2}] \\
c_{16}(t) &= \frac{1}{6} H_{1jk} [\rho_{12} \rho_{j4} \rho_{k4} + \rho_{14} \rho_{j2} \rho_{k4} + \rho_{14} \rho_{j4} \rho_{k2}] \\
c_{17}(t) &= \frac{1}{6} H_{1jk} \rho_{13} \rho_{j3} \rho_{k3} \\
c_{18}(t) &= \frac{1}{6} H_{1jk} [\rho_{13} \rho_{j3} \rho_{k4} + \rho_{13} \rho_{j4} \rho_{k3} + \rho_{14} \rho_{j3} \rho_{k3}] \\
c_{19}(t) &= \frac{1}{6} H_{1jk} [\rho_{13} \rho_{j4} \rho_{k4} + \rho_{14} \rho_{j3} \rho_{k4} + \rho_{14} \rho_{j4} \rho_{k3}] \\
c_{20}(t) &= \frac{1}{6} H_{1jk} \rho_{14} \rho_{j4} \rho_{k4}
\end{aligned} \tag{75}$$

Here, the variable ρ_{mn} is the element in the m th row and the n th column of the transformation matrix, $\mathbf{E}$. The periodic $\mathbf{X}$

and $\mathbf{E}$ will again be integrated over one period, but for this representation, at each of the evenly spaced samplings, the expanded modal coefficients, c_i , will be calculated. Like the sampled sets in the exact representation, each of the sampled sets of c_i will be converted into a set of one hundred Fourier coefficients.

Starting with the initial modal variables, found in the exact solution at $t=0$, the expanded Hamiltonian is integrated over the same time as the exact solution. The results can be plotted as the expanded solution and can be compared to the exact solution. The modal equations of motion come from the expanded Hamiltonian and are

$$\dot{y}_1 = \frac{\partial K}{\partial y_2} = \frac{\partial (K_2)}{\partial y_2} + c_2 y_1^2 + 2c_5 y_1 y_2 + c_6 y_1 y_3 + c_7 y_1 y_4 + 3c_{11} y_2^2 + 2c_{12} y_2 y_3 + 2c_{13} y_2 y_4 + c_{14} y_3^2 + c_{15} y_3 y_4 + c_{16} y_4^2 \quad (76)$$

$$\dot{y}_2 = -\frac{\partial K}{\partial y_1} = -\frac{\partial (K_2)}{\partial y_1} - 3c_1 y_1^2 - 2c_2 y_1 y_2 - 2c_3 y_1 y_3 - 2c_4 y_1 y_4 - c_5 y_2^2 - c_6 y_2 y_3 - c_7 y_2 y_4 - c_8 y_3^2 - c_9 y_3 y_4 - c_{10} y_4^2 \quad (77)$$

$$\dot{y}_3 = \frac{\partial K}{\partial y_4} = \frac{\partial (K_2)}{\partial y_4} + c_4 y_1^2 + c_7 y_1 y_2 + c_9 y_1 y_3 + 2c_{10} y_1 y_4 + c_{13} y_2^2 + c_{15} y_2 y_3 + 2c_{16} y_2 y_4 + c_{18} y_3^2 + 2c_{19} y_3 y_4 + 3c_{20} y_4^2 \quad (78)$$

$$\dot{y}_4 = -\frac{\partial K}{\partial y_3} = -\frac{\partial (K2)}{\partial y_3} - c_3 y_1^2 - c_6 y_1 y_2 - 2c_8 y_1 y_3 - c_9 y_1 y_4 - c_{12} y_2^2 - 2c_{14} y_2 y_3 - c_{15} y_2 y_4 - 3c_{17} y_3^2 - 2c_{18} y_3 y_4 - c_{19} y_4^2 \quad (79)$$

where the term $\partial(K2)/\partial y_1$ depends on the form of **S** in the quadratic term of the modal Hamiltonian. Appendix A also defines the forms of $\partial(K2)/\partial y_1$ for the various types of eigenvector pairs.

IV. Software

The software programs used in this study are coded in Fortran 77. All seven main programs and eleven subroutines can be found in Appendix B. The majority of the programs were written with the restricted three-body problem in mind, but any of the software can easily be modified to any periodic Hamiltonian system.

4.1 *Surface of Section*

The first program used in the study is the program SECTION. The core of the program was written by Ross (1991) with the purpose of duplicating the surface of section plots created by Jefferys (1971). The program is written to integrate several initial conditions, while storing the apoapsis and periapsis crossings in a plot file. While the program is simple in its concept, finding initial conditions that produce periodic regions in the phase space is guesswork at best. Validation of this program is accomplished through direct comparison with the atlas of surface of sections created by Jefferys.

4.2 *Determination of Periodic Initial Conditions*

Once the surface of section plot has been created, determination of exact initial conditions for a periodic trajectory must be found. The reader should recall that this was done by selecting an initial condition where the

trajectory was crossing the q_1 axis at the initial time. Therefore the initial condition on q_2 is zero; the guessed initial condition on q_1 is found by estimating the center of a periodic region on the q_1 axis. The third initial condition, the period, is estimated by examining the time it takes for a nearby orbit to nearly return to its initial location. These estimated initial conditions are loaded into the program PERIOD which iteratively integrates and corrects q_1 and τ until at the end of one period, the trajectory has returned to the initial q_1 and q_2 position. A benefit of this program is that it is self checking, it accomplishes this by forcing the final conditions and initial conditions to match. Verification of the coded equations of motion is the only check needed for this program, and this is best done by hand.

Once the initial conditions are found, the program extracts the eigenvalues, eigenvectors, and Poincaré exponents of the system.

4.3 *Symplectic Normalization*

The type of symplectic normalization required is determined by examination of the types of eigenvector and Poincaré exponent pairs found in program PERIOD. Some rearrangement of the eigenvectors and eigenvalues may be required in order to ensure positive/negative or degenerate pairs are kept together. Once properly arranged, the eigenvectors are fed into program EXSYRL (EXTENDED,

SYMPLECTIC, REAL). The purpose of this program is to find an extended eigenvector if needed, convert the standard eigenvectors into symplectic eigenvectors, and ensure that the symplectic eigenvectors and the $\mathbf{J}$ matrix are real valued. The best check of this program is to ensure the final forms of $\mathbf{E}$ and $\mathbf{J}$ satisfy Eq(11) at several different times over the period of motion.

4.4 Storage of the Periodic Trajectory and Hamiltonian Coefficients

The final $\mathbf{E}$ and $\mathbf{J}$ matrices are loaded into programs FLOQUET and HAMILTONIAN to create the sets of Fourier coefficients for either the exact or the expanded perturbation problems.

FLOQUET integrates the periodic orbit and extracts two hundred evenly spaced values of the state vector and the symplectic eigenvector matrix. These sets are then fed to subroutine FOURIER which converts them into one hundred pairs of Fourier coefficients allowing the periodic trajectory to be reformed at any time.

HAMILTONIAN also integrates the periodic trajectory and extracts the state vector and the $\mathbf{E}(t)$ matrix values at two hundred evenly spaced points on the trajectory. It then performs a Taylor's series expansion of the Hamiltonian to find the twenty Hamiltonian coefficients, $c_i(t)$, at each of the two hundred points. These sets of $c_i(t)$ are also loaded into FOURIER to be converted into one hundred pairs of

Fourier coefficients for each set of $c_1(t)$.

One check for these two programs is to ensure that the state vector and $\mathbf{E}$ have returned to their initial condition after one period. This should not be a problem since this same check is performed in PERIOD. A valuable check of the Fourier coefficients is performed by summing the Fourier cosines. Each cosine sum should equal the initial condition of the function the Fourier coefficients describe (Brower and Clemence, 1961:110).

4.5 *The Perturbation Solutions*

The exact and the expanded perturbation problems are analyzed by programs of the same names. Program EXACT takes the Fourier coefficients created in FLOQUET and reassembles the periodic trajectory conditions on demand. Perturbed initial conditions are integrated in time, and at regular intervals the state vector is extracted. The periodic state is subtracted from the perturbed state and the result is converted into the modal variables by the corresponding $\mathbf{E}(t)$.

Program EXPANDED requires not only the Fourier coefficients created in program HAMILTONIAN, but also the initial modal conditions as found by program EXACT. EXPANDED then integrates the modal variables directly. Both programs extract the modal variables at regular intervals and send the results to plot files. If the two programs are working correctly, there should be a certain amount of

correspondence of the plots for the two solutions. The actual amount of correlation is a topic for the next chapter.

V. Results and Discussions

5.1 Canonical Floquet Theory

The most significant result of the canonical Floquet theory is that it works. The application of canonical Floquet theory is currently limited to Hamiltonian systems. There appears to be no limits to the degrees of freedom that can be involved as long as the system can be written as a periodic Hamiltonian. This now opens the doors to a great number of possibilities for perturbation solutions to periodic systems. Thus, the results of the three-body perturbation solution will be of more interest than any further discussion of the theory.

5.2 The Three-Body Perturbation Problem

Recall, the particular three-body system of interest in this study is the Sun-Jupiter system; this means a $\mu = 0.00095388$. The surface of section for this system was presented in Figure 4. The first task was to find a periodic region in this phase space. The surface of section was created using a Jacobian constant of $J = 3.15$ and a dozen initial state conditions along the x axis. In examining the x axis between the two primary masses (i.e. between -1 and 0), there appears to be two main periodic regions; one centered at approximately $x = -0.3$ and the other at approximately $x = -0.5$ on the surface of section.

The five crescent shaped structures in between the first and second central rings define the periodic region associated with $x = -0.3$, while there are four outer crescents that define the periodic region associated with $x = -0.5$. Figure 5 shows a close-up view of the $x = -0.5$ region and provides an initial guess of $x = -0.51$ for the periodic initial condition, the center of the crescent structure. After iteration of the guessed initial conditions with program PERIOD, the actual initial conditions are determined to be $x = -0.5136620321$, $y = 0.0$, and a period of 6.256580411 (all rounded to ten significant figures) for the given Jacobian and μ . Converting these conditions into the initial conditions for Szebehely's equations yields $q_1 = -0.5127081521$, $q_2 = 0.0$, and $H = -1.574523515$, with the same period of motion. This particular periodic trajectory produced a positive/negative imaginary pair of Poincaré exponents as well as a degenerate pair, making it a good test case for the canonical Floquet theory as well as the perturbation analysis.

5.2.1 *Unperturbed System*

The starting point for analysis in the study was to examine the time history of the modal vectors for the unperturbed case. Since an imaginary pair of ω_1 produces a sine and a cosine function in the modal variables, a plot of one versus the other should produce a circle. For the unperturbed case, the modal vectors should equal zero for

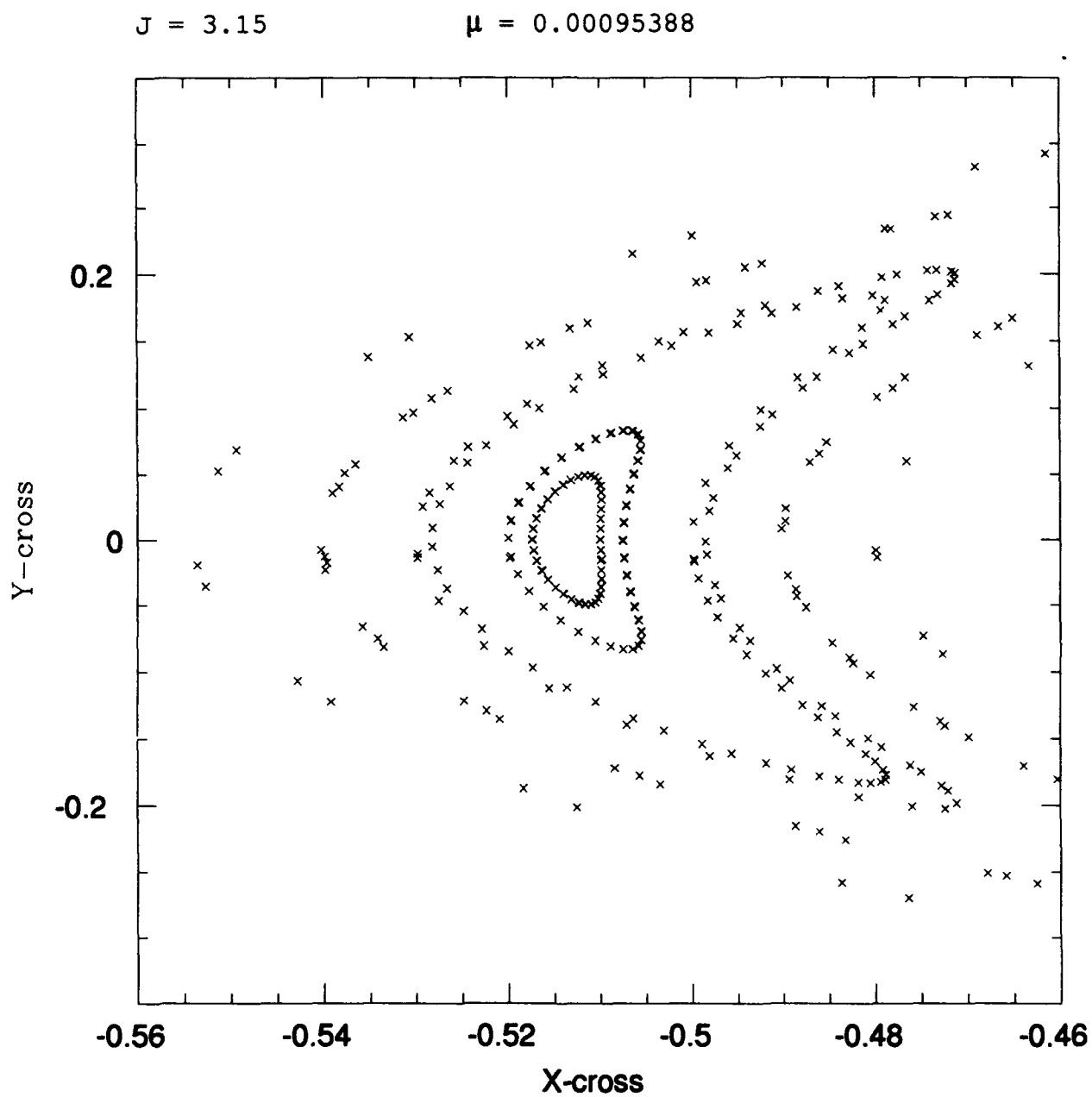


Figure 5: Close-up View of Periodic Region of the Sun-Jupiter Surface of Section

all time. Figure 6 shows the first and second modal vectors plotted against each other. The other two modal vectors are simply plotted against time in Figures 7 and 8. In all three figures, the scale is such that maximum deviation will be seen, but in all cases the magnitude of the deviation is zero for purposes of this analysis. Looking at the deviations in this detail do show trends in the data that can be compared to data in the perturbation cases. The fact that any error shows at all is almost certainly due to truncation error in all the calculations made on this data.

5.2.2 Perturbed Systems

For this study only two initial values were perturbed for examination, first q_1 and then J ; J can be translated into a change in H . Each quantity affected the value of p_2 through the equations of motion, but no other initial conditions are changed. Physically, a change in q_1 , and p_2 , means a change in position on the x axis of the system in Figure 1 along with a change in the y velocity, p_2 , in order to stay on the same Hamiltonian surface, H . The change in the Hamiltonian value allows for the position to remain the same, but a change in the y velocity is still required.

5.2.2.1 Changes in q_1

As in any perturbation problem, the magnitude of the change must be small in order to keep the truncated, expanded Hamiltonian valid. Since the distance between the two primary masses has been non-dimensionalized to a value

$J = 3.15$
 $\mu = 0.00095388$

$\delta J = 0.0$
 $\delta q_1 = 0.0$

— Exact
x x x Expanded

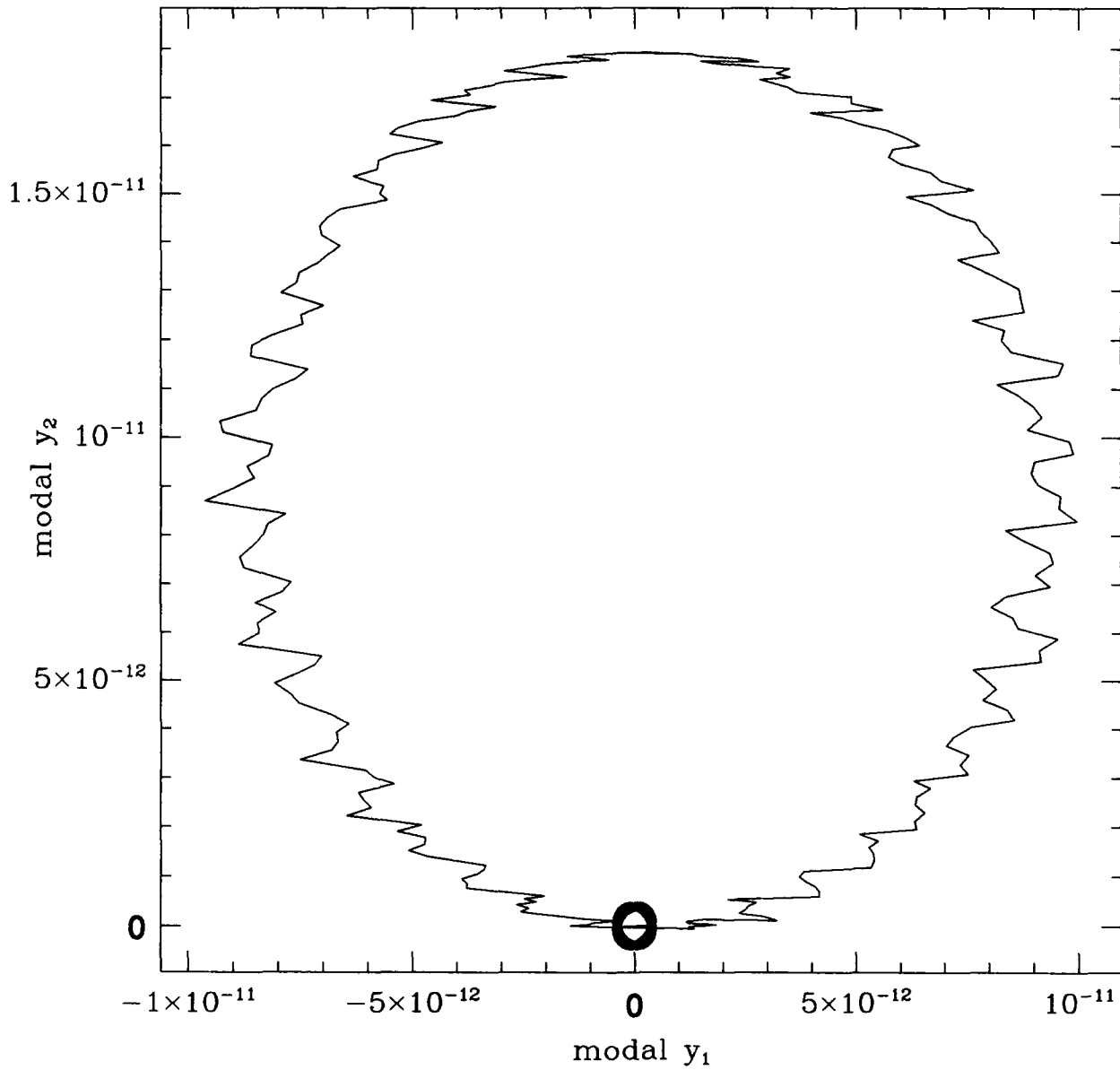


Figure 6: Unperturbed Oscillatory Modal Variables, y_1 versus y_2

$J = 3.15$
 $\mu = 0.00095388$

$\delta J = 0.0$
 $\delta q_1 = 0.0$

— Exact
x x x Expanded

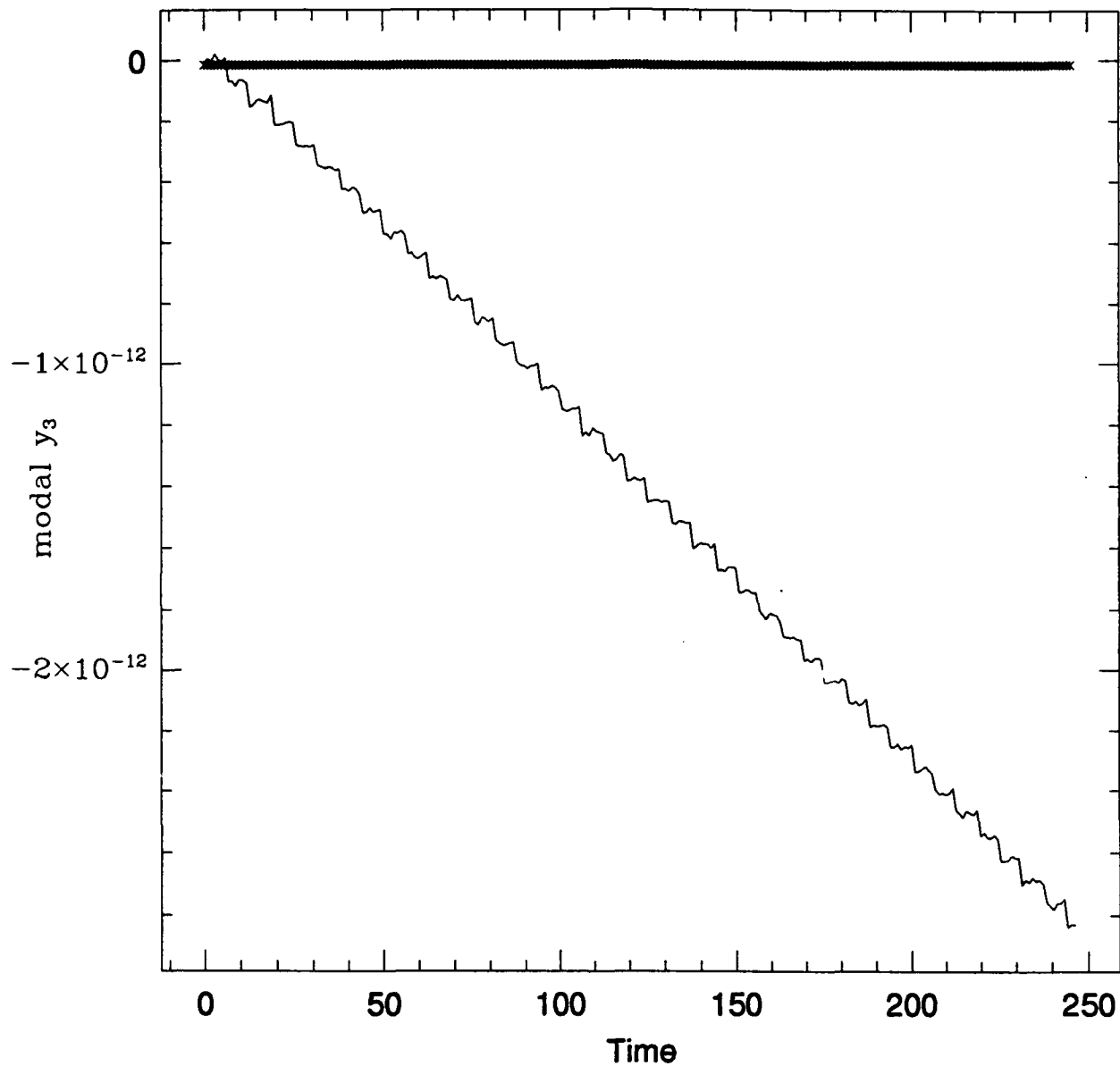


Figure 7: Unperturbed Time History of Modal Variable y_3

$J = 3.15$
 $\mu = 0.00095388$

$\delta J = 0.0$
 $\delta q_1 = 0.0$

— Exact
x x x Expanded

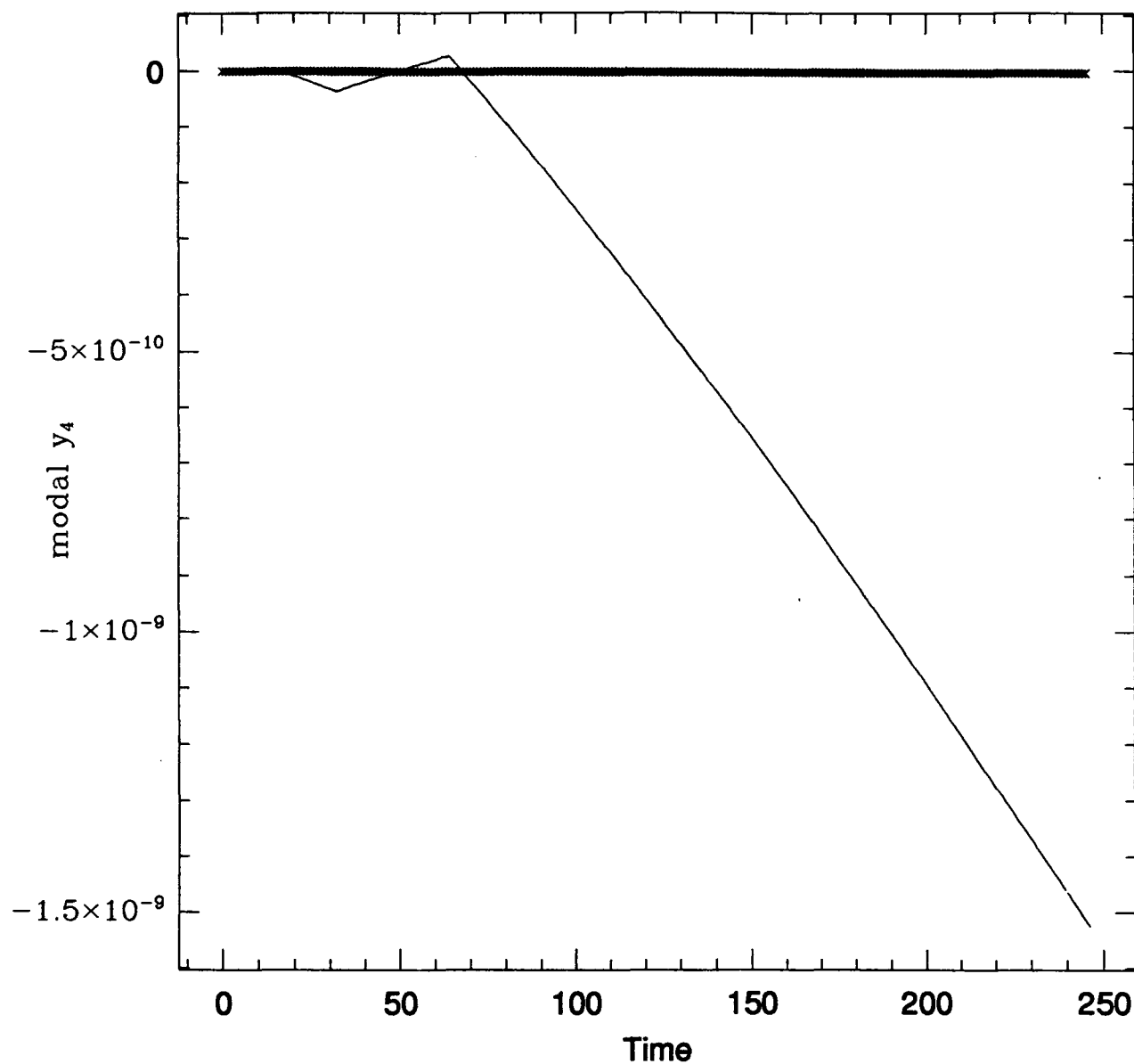


Figure 8: Unperturbed Time History of Modal Variable y_4

of one, it would be accurate to say that a perturbation of 0.001 for the third body in the system, or a tenth of a percent, is significant. In physical dimensions this would be an approximate position change of 775,000 km, or more than twice the distance to the moon from earth. The initial change to the position was chosen to be $\delta q_1 = 1e-8$, or about 7.75 km. It can be seen in Figure 9 that the perturbed solution of the oscillatory modal variables describes a visually perfect circle. Some particular items of interest: 1) the circle does close on itself as can be seen by an overlapping region near the left side of the plot, 2) both the exact and the expanded solution match precisely for this initial displacement, 3) the amplitude of the y_1 and y_2 modal variables is about three times the initial position displacement, and 4) a time history of the first two modal variables, Figures 10 and 11, shows that these two variables do in fact describe smooth sine and cosine functions at this level of displacement. The time history for the perturbation in modals y_3 and y_4 is not presented at this point because the magnitudes of the perturbations is still nearly zero. The exact and expanded solutions of y_3 and y_4 match perfectly at $\delta q_1 = 1e-8$.

The magnitude of the perturbation is then increased to $\delta q_1 = 1e-6$, or about 775 km (the distance from San Diego to San Francisco). Although this is an enormous distance for an Earth satellite to be out of orbit, is it a significant

$J = 3.15$
 $\mu = 0.00095388$

$\delta J = 0.0$
 $\delta q_1 = 1.0e-8$

— Exact
x x x Expanded

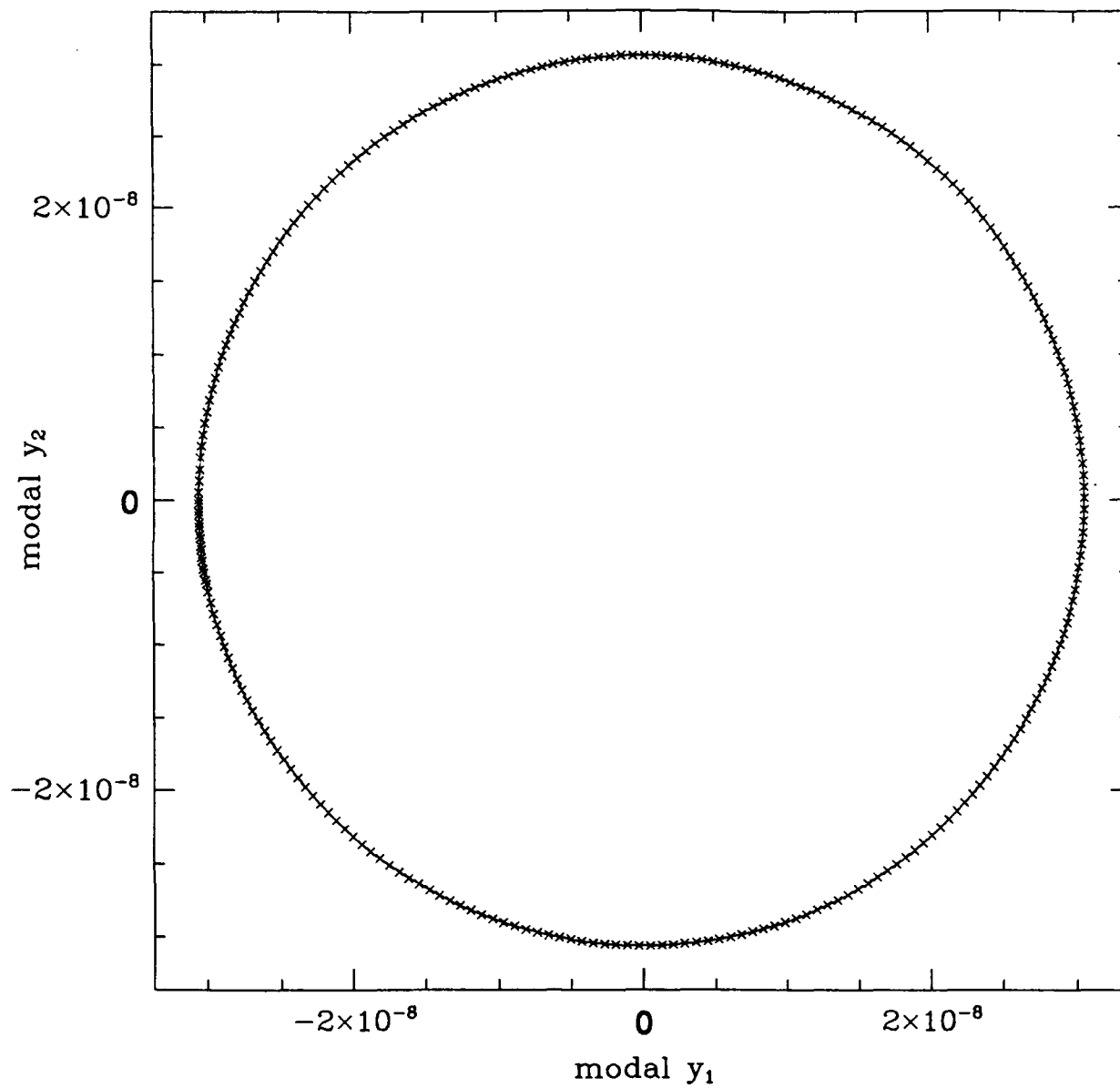


Figure 9: Oscillatory Modal Variables,
 y_1 versus y_2 , $\delta q_1 = 1e-8$

$J = 3.15$
 $\mu = 0.00095388$

$\delta J = 0.0$
 $\delta q_1 = 1.0e-8$

— Exact
x x x Expanded

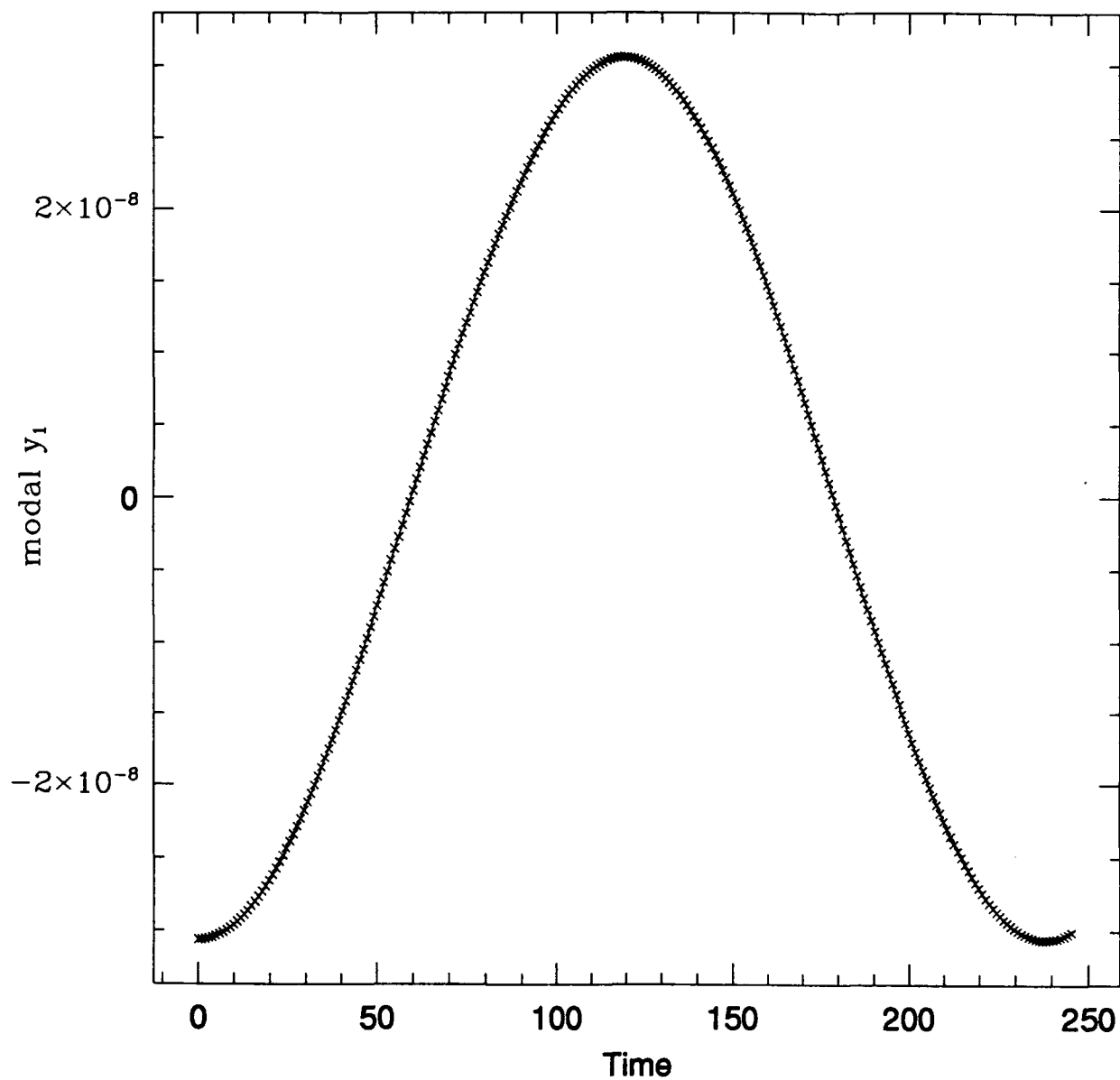


Figure 10: Time History of Modal
Variable y_1 , $\delta q_1 = 1e-8$

$J = 3.15$
 $\mu = 0.00095388$

$\delta J = 0.0$
 $\delta q_1 = 1.0e-8$

— Exact
x x x Expanded

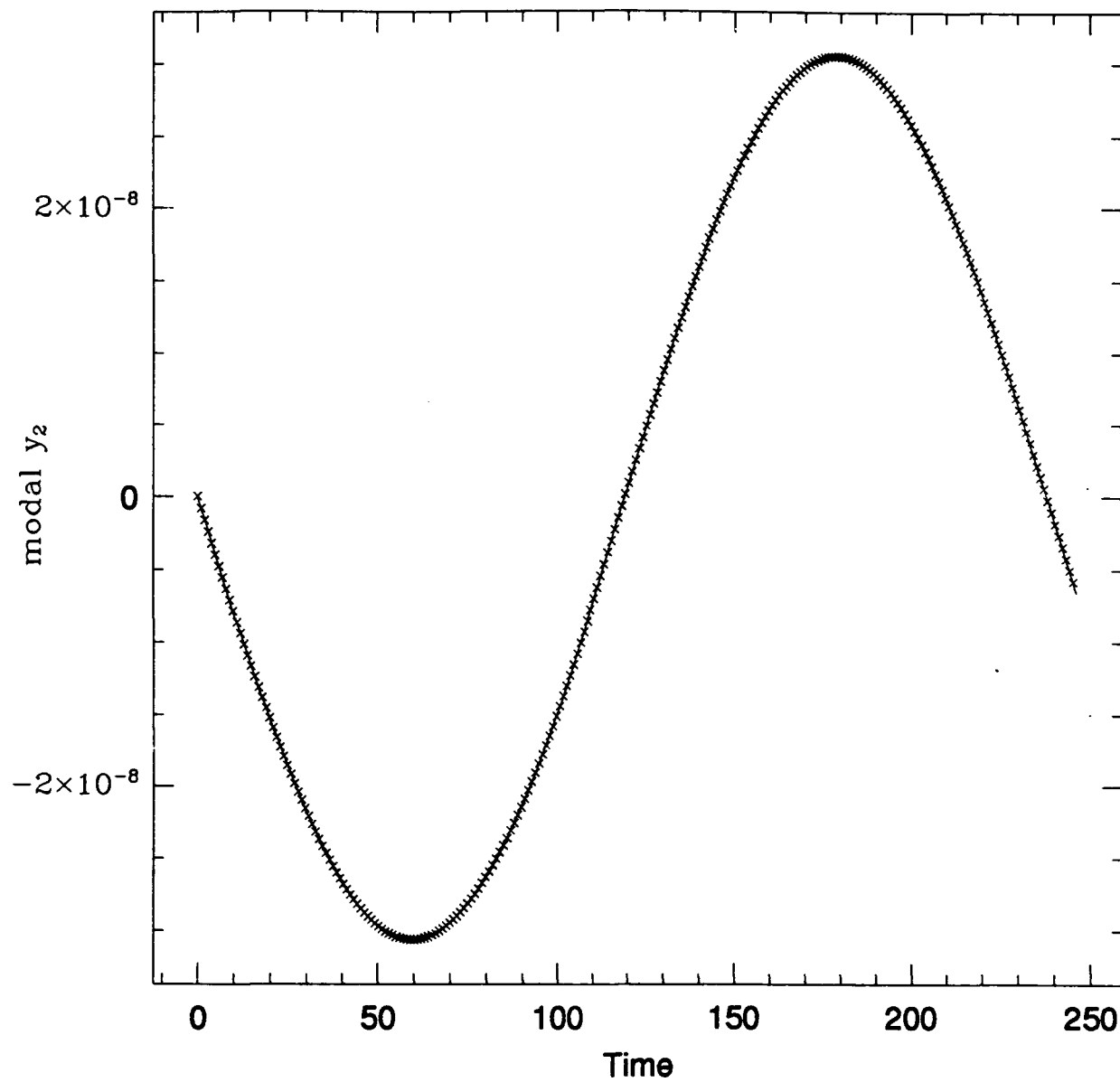


Figure 11: Time History of Modal
Variable y_2 , $\delta q_1 = 1e-8$

perturbation for a periodic trajectory in the Sun-Jupiter system? Figure 12 shows that although there is a slight waver in the circular path of the first two modals, it is still very close to cyclic behavior. The time histories of the first two modals show no difference to the sine and cosine functions of Figures 10 and 11. Note also that the amplitude of these modals is still approximately three times that of the initial displacements and the exact and expanded solutions still follow each other perfectly. At this point, the magnitude of the third and fourth modals (Figures 13 and 14) are becoming more significant, but they are still at least one hundred times smaller than the first two modals. It is interesting to note how closely the exact and expanded solutions match in all modals at this point.

Increasing the initial displacement even further, to $\delta q_1 = 5e-5$, or about 39,000 km (three times the diameter of the earth), a drastic change is found in not only the modal response, but also the correlation of the exact and expanded solutions. As can be seen in Figure 15, although the exact solution does close on itself, it follows a very erratic path that quickly loses its circular appearance. The jagged, sharp changes in many of the figures at this point are partially do to the large steps between points in the plot, but most of the problem is occurring because the periodic trajectory is dissolving. This can be seen in Figure 16 where the plot of the oscillatory modals in the

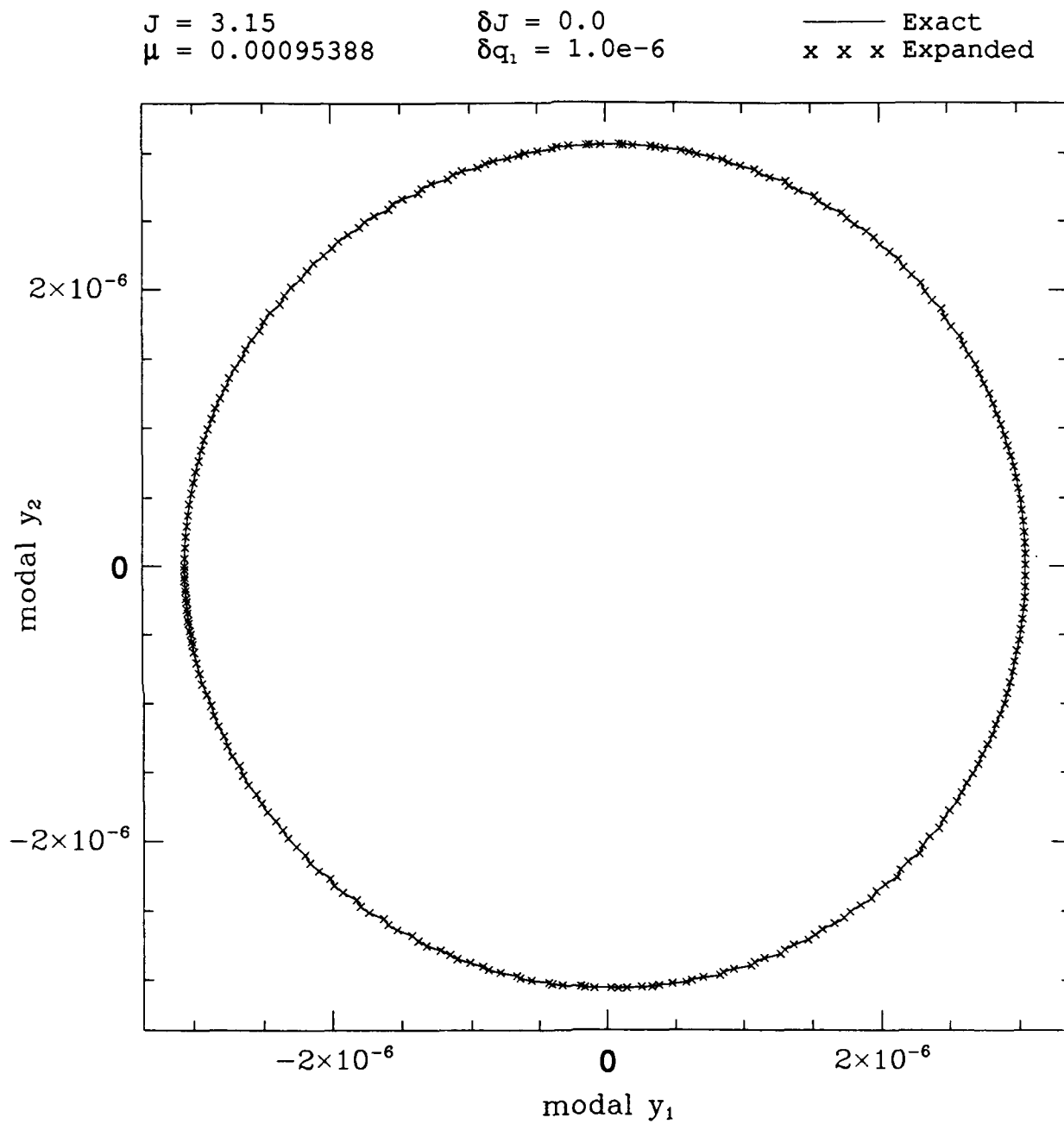


Figure 12: Oscillatory Modal Variables,
 y_1 versus y_2 , $\delta q_1 = 1e-6$

$J = 3.15$
 $\mu = 0.00095388$

$\delta J = 0.0$
 $\delta q_1 = 1.0e-6$

— Exact
x x x Expanded

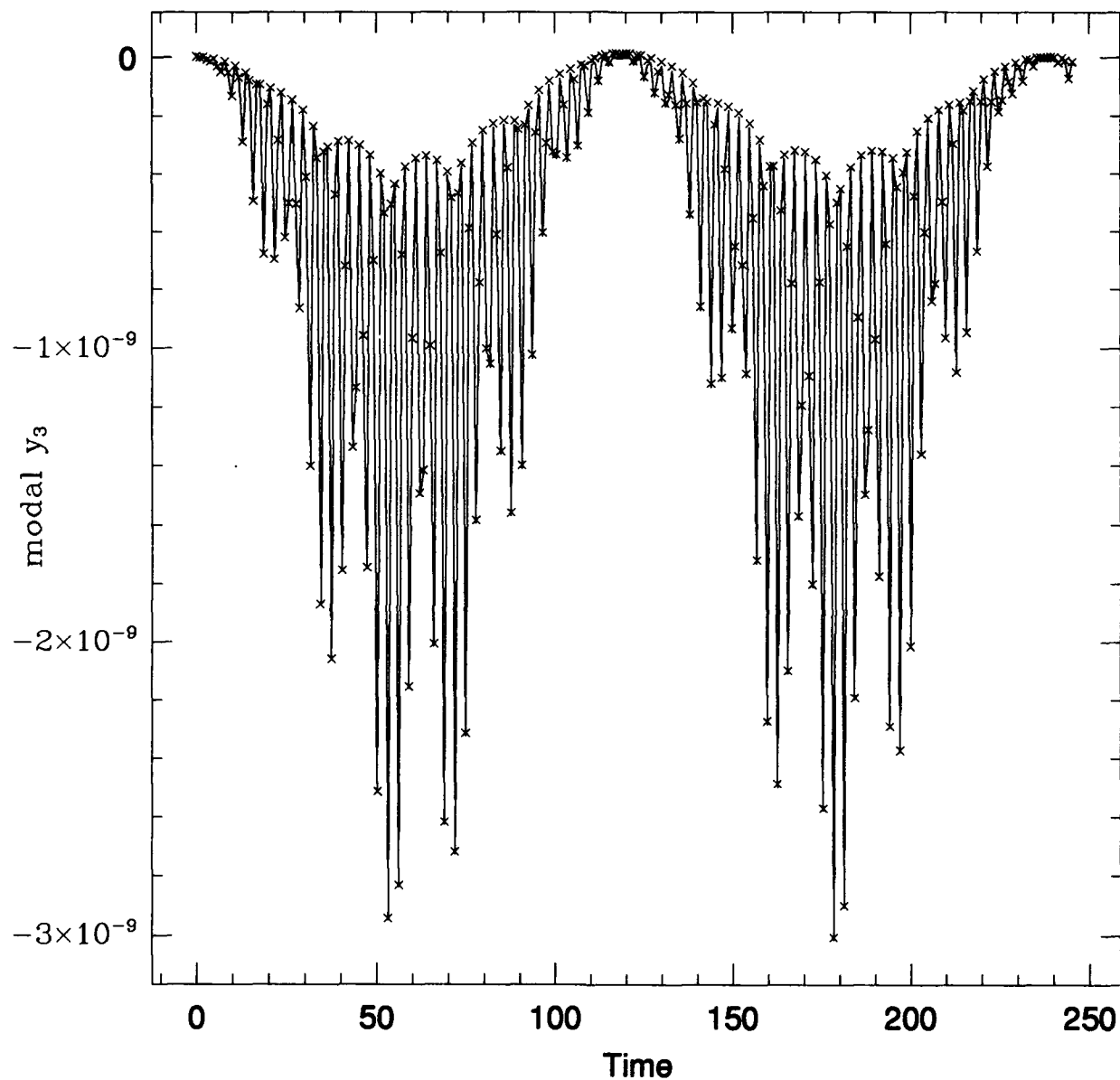


Figure 13: Time History of Modal Variable y_3 , $\delta q_1 = 1e-6$

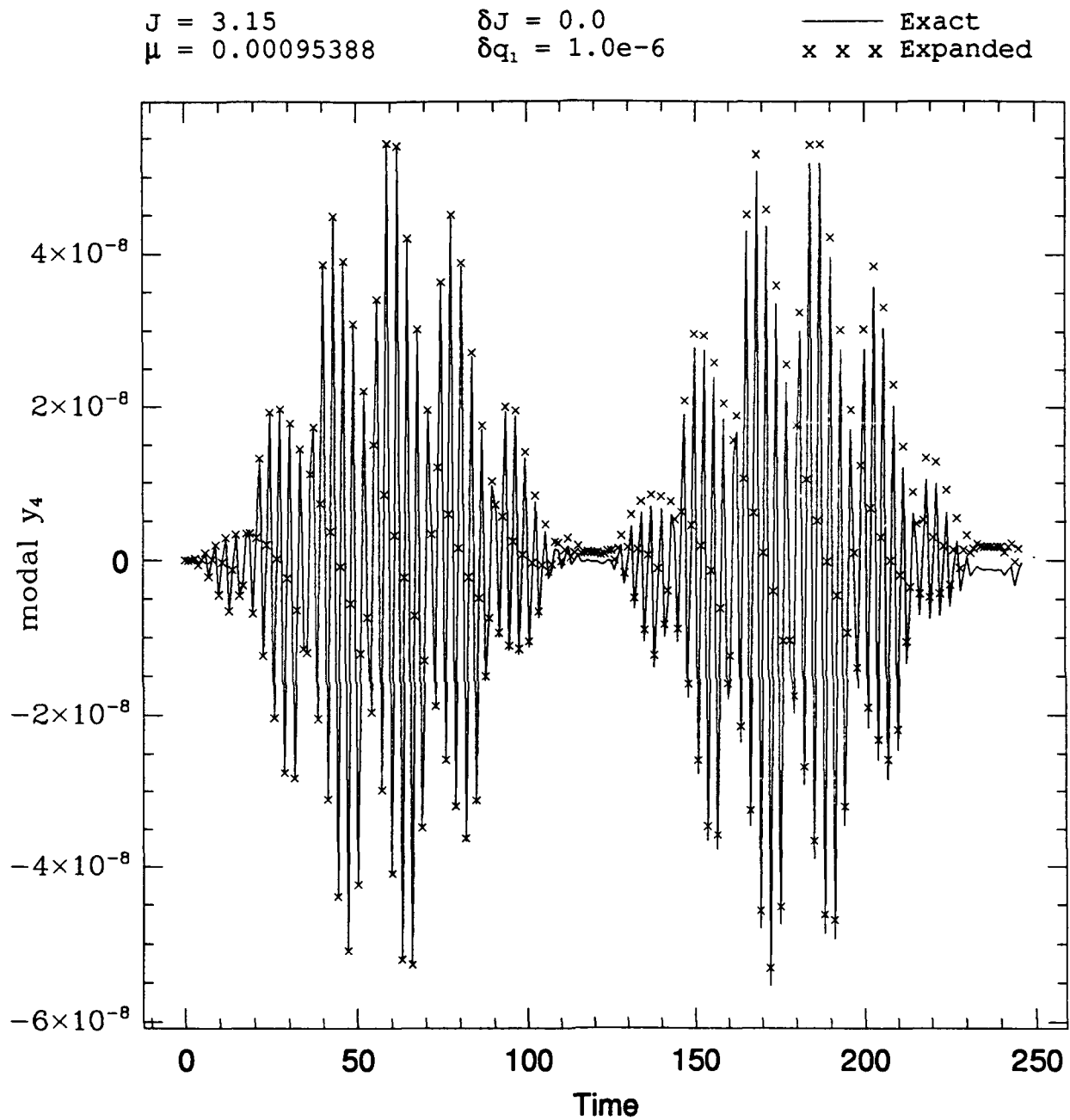


Figure 14: Time History of Modal Variable y_4 , $\delta q_1 = 1e-6$

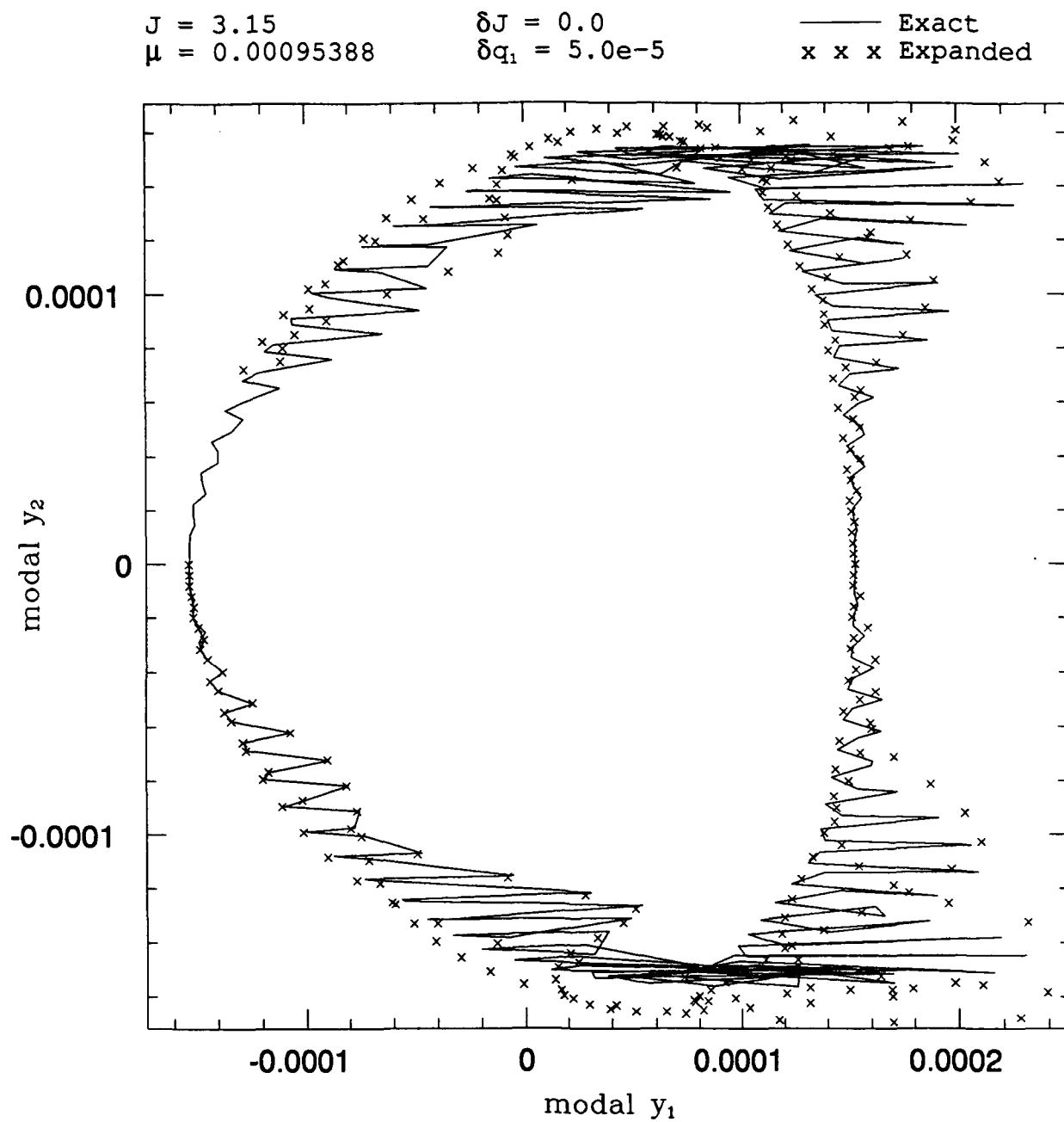


Figure 15: Oscillatory Modal Variables,
 y_1 versus y_2 , $\delta q_1 = 5e-5$

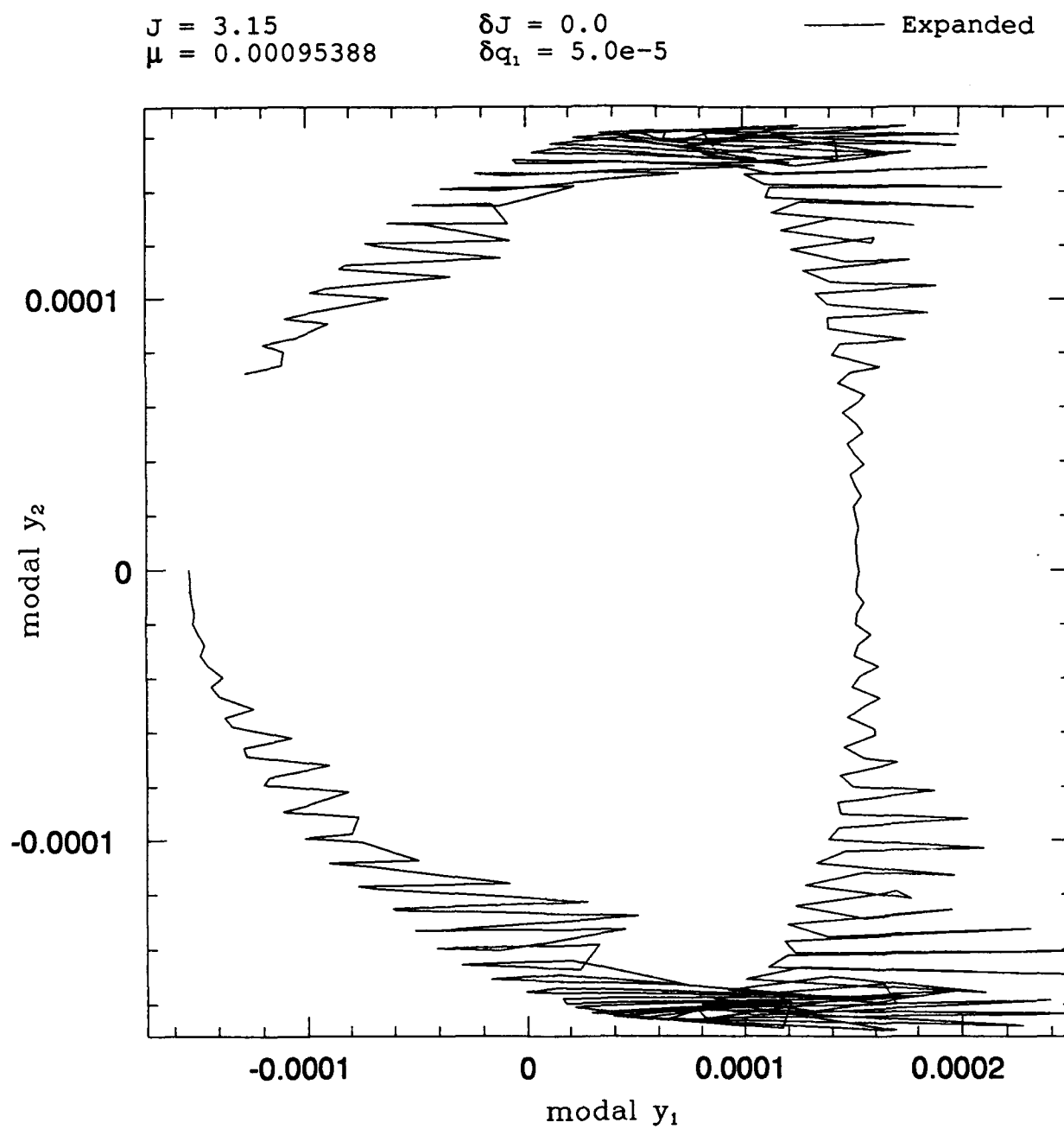


Figure 16: Oscillatory Modal Variables, y_1 versus y_2 , Expanded Solution, $\delta q_1 = 5e-5$

expanded solution is shown alone. The integration time is no longer enough to allow these modal variables to close on themselves. The time histories of the four modal variables for both the exact and the expanded solutions are shown in Figures 17 to 21. In every case, the expanded solution appears to be lagging the exact solution. A larger deviation in form also starts showing up in modals y_3 and y_4 of Figures 20 and 21. It appears obvious that even at this point, the expanded solution is no longer accurate. The periodic nature of the trajectory does seem to hold for the exact solution, but with such erratic behavior over part of the time history it is doubtful that the third-body in this system could maintain its trajectory for long.

Finally, as the change to q_1 continues to increase, the expanded solution loses all its coherence first, as can be seen in Figure 22, while the exact solution becomes entirely distorted by $\delta q_1 = 1e-3$ in Figure 23. Looking more closely at the first two modal variables when $\delta q_1 = 1e-3$ (twice the distance to the moon), it can be seen that the y_1 modal has completely lost the cosine shape and looks a great deal like the y_3 modal inverted. The y_2 modal has lost the smooth sine shape and appears to be approaching the same shape as the y_4 modal (see Figures 24 and 25). The shape of the y_3 and y_4 modals has not changed since the initial perturbation, while the magnitude has increased with the perturbations. Meanwhile, the expanded solution for the y_3

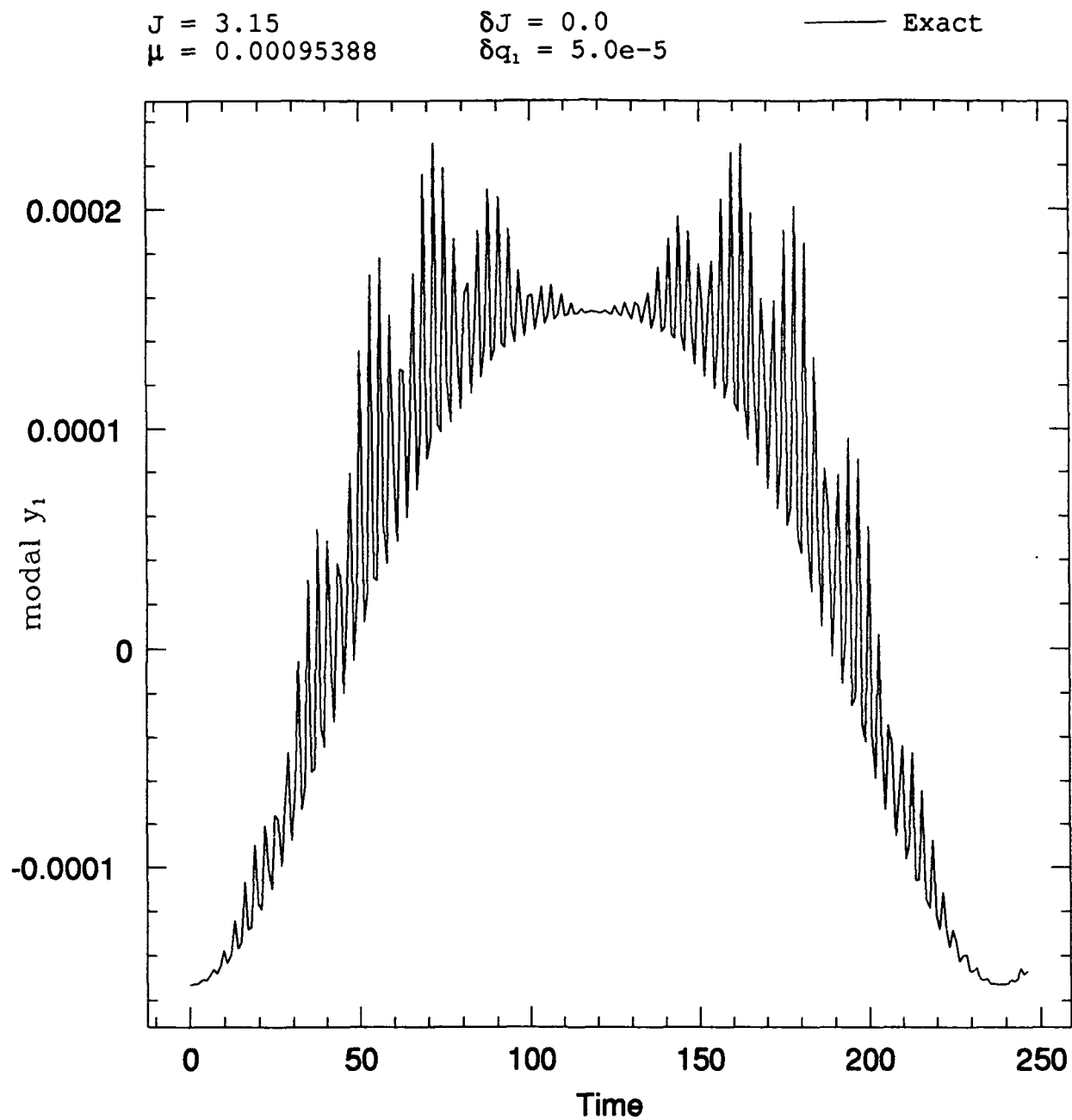


Figure 17: Time History of Modal Variable y_1 , Exact Solution, $\delta q_1 = 5e-5$

$J = 3.15$
 $\mu = 0.00095388$

$\delta J = 0.0$
 $\delta q_1 = 5.0e-5$

— Expanded

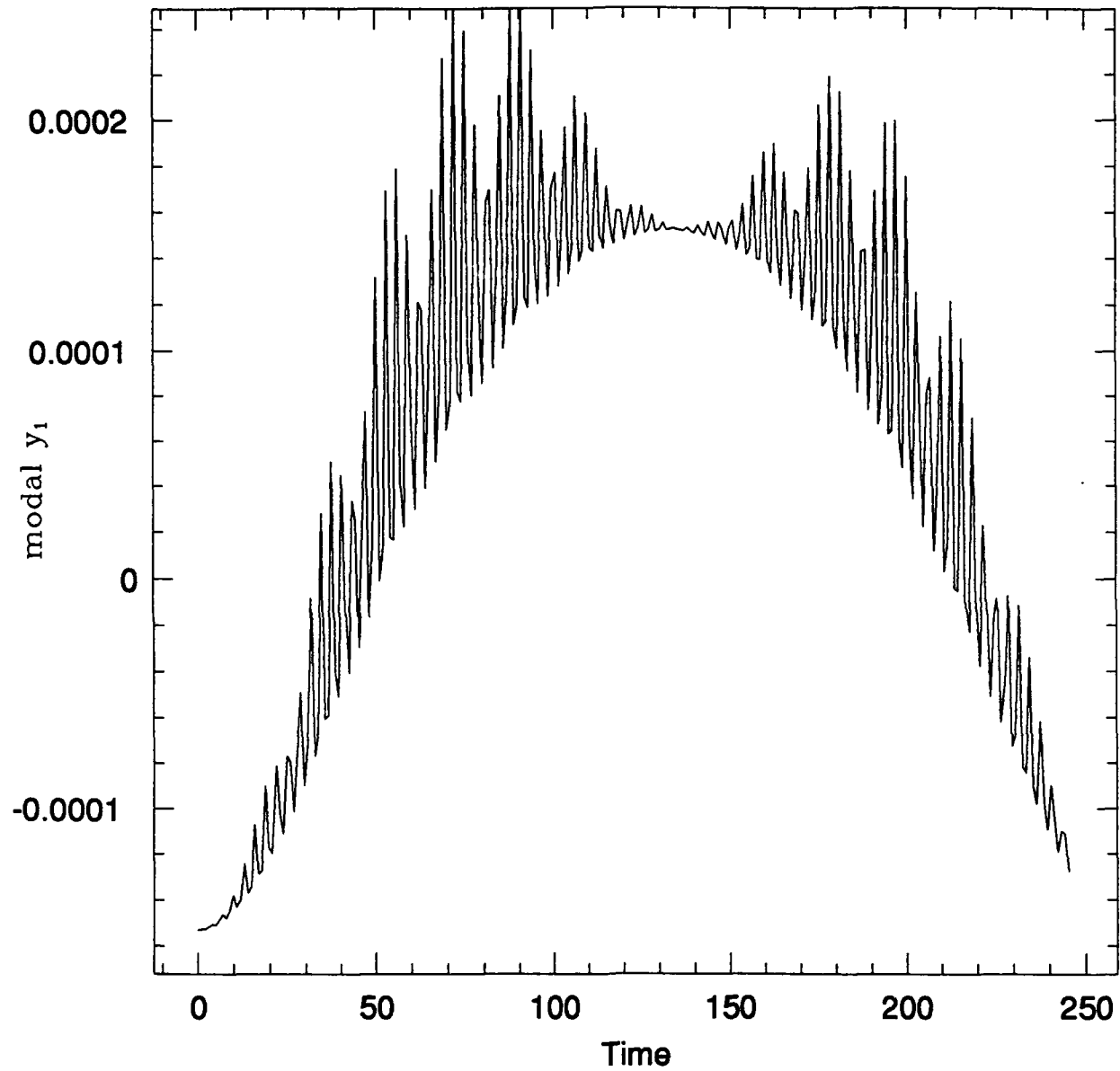


Figure 18: Time History of Modal Variable y_1 , Expanded Solution, $\delta q_1 = 5e-5$

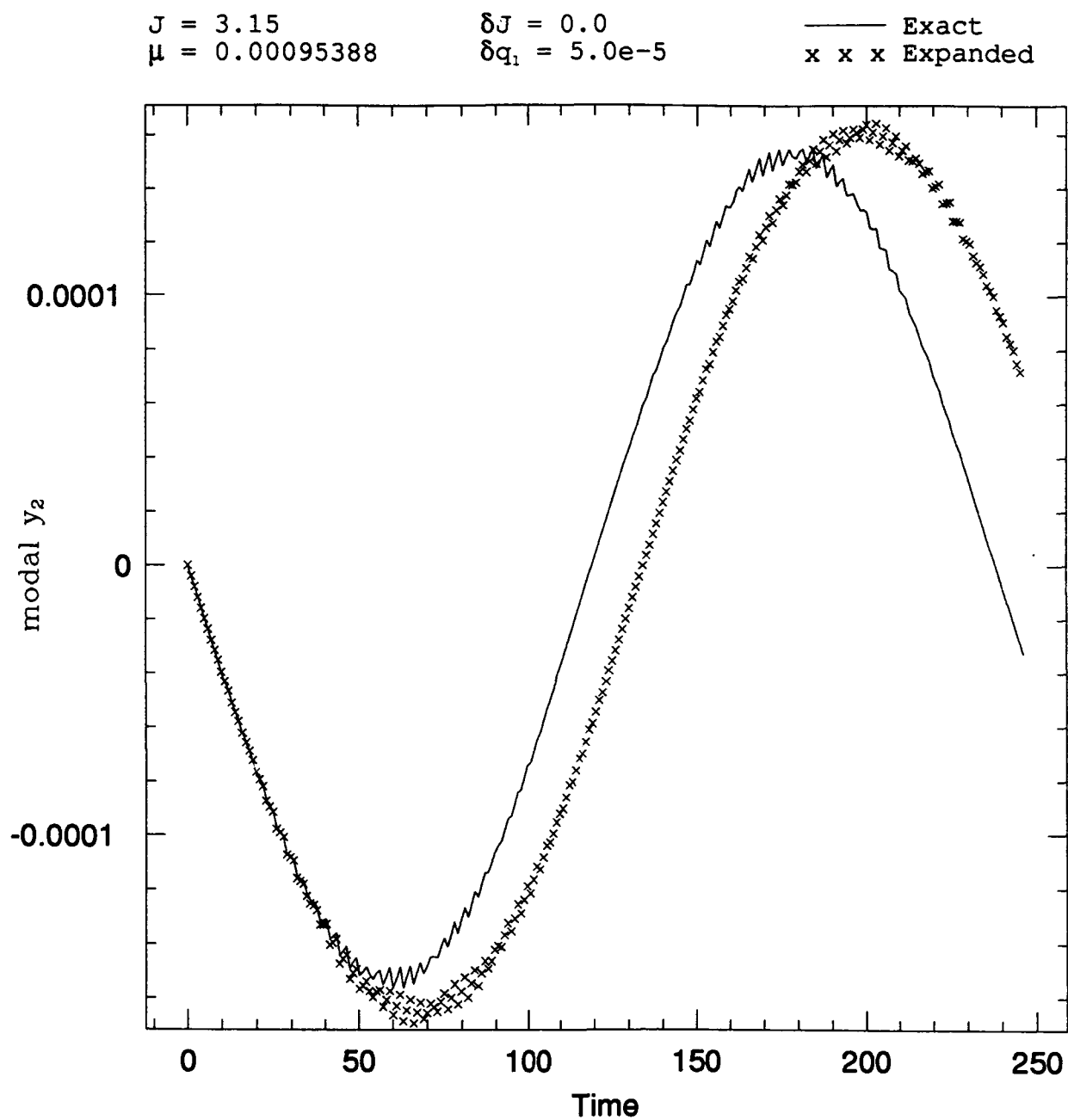


Figure 19: Time History of Modal Variable y_2 , $\delta q_1 = 5e-5$

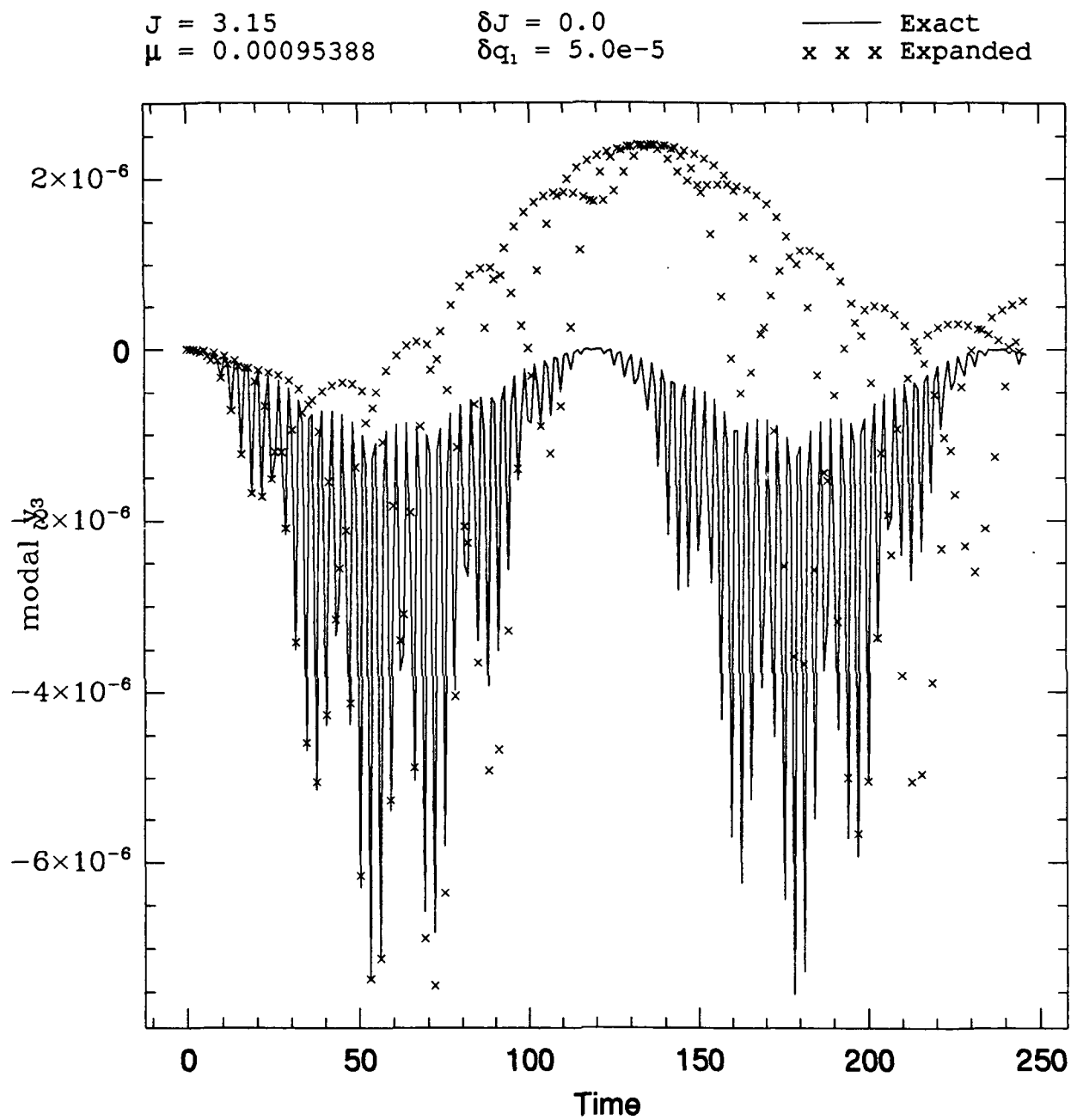


Figure 20: Time History of Modal Variable y_3 , $\delta q_1 = 5e-5$

$J = 3.15$
 $\mu = 0.00095388$

$\delta J = 0.0$
 $\delta q_1 = 5.0e-5$

— Exact
 x x x Expanded

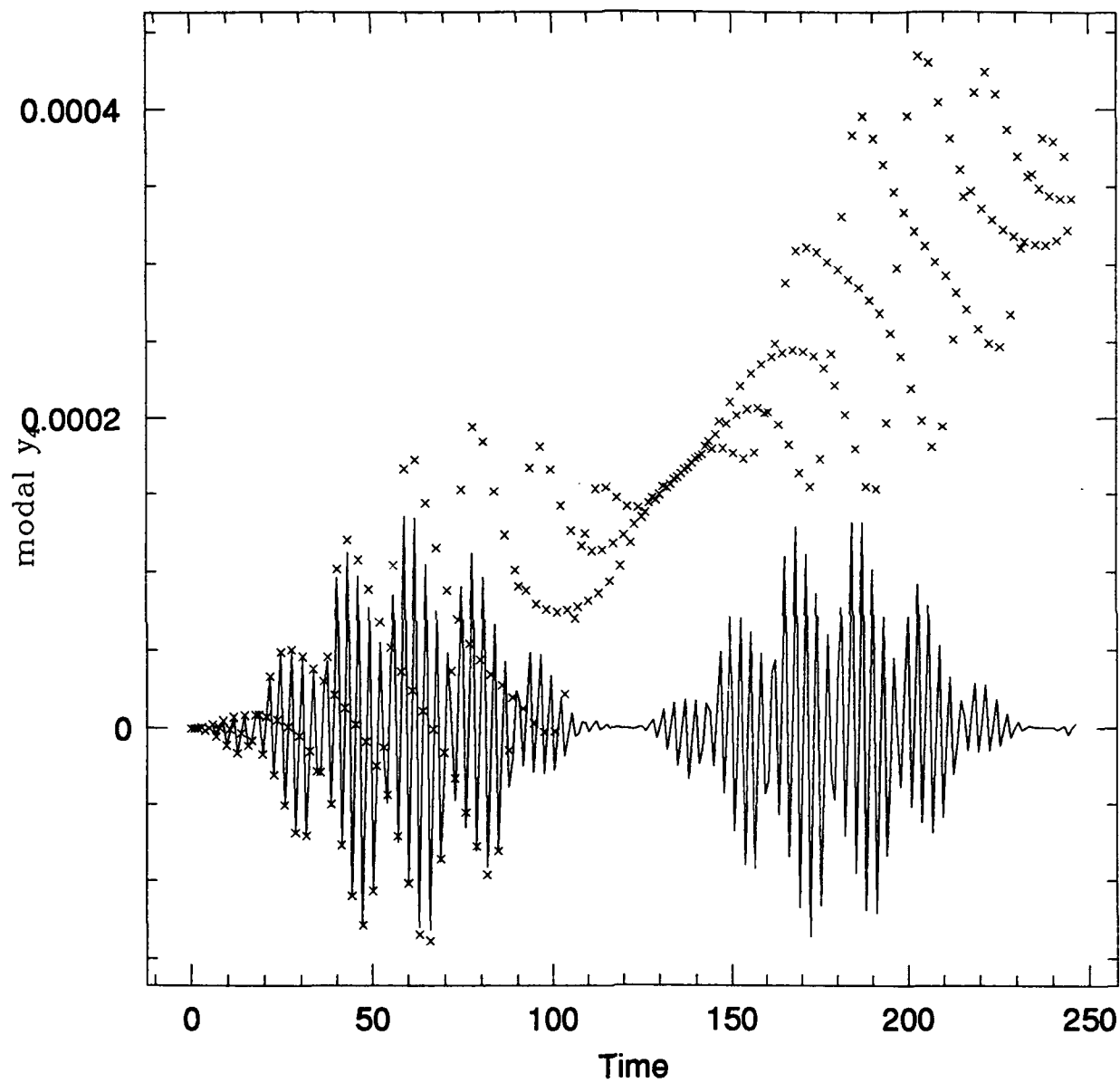


Figure 21: Time History of Modal
 Variable y_4 , $\delta q_1 = 5e-5$

$J = 3.15$
 $\mu = 0.00095388$

$\delta J = 0.0$
 $\delta q_1 = 1.0e-4$

— Exact
x x x Expanded

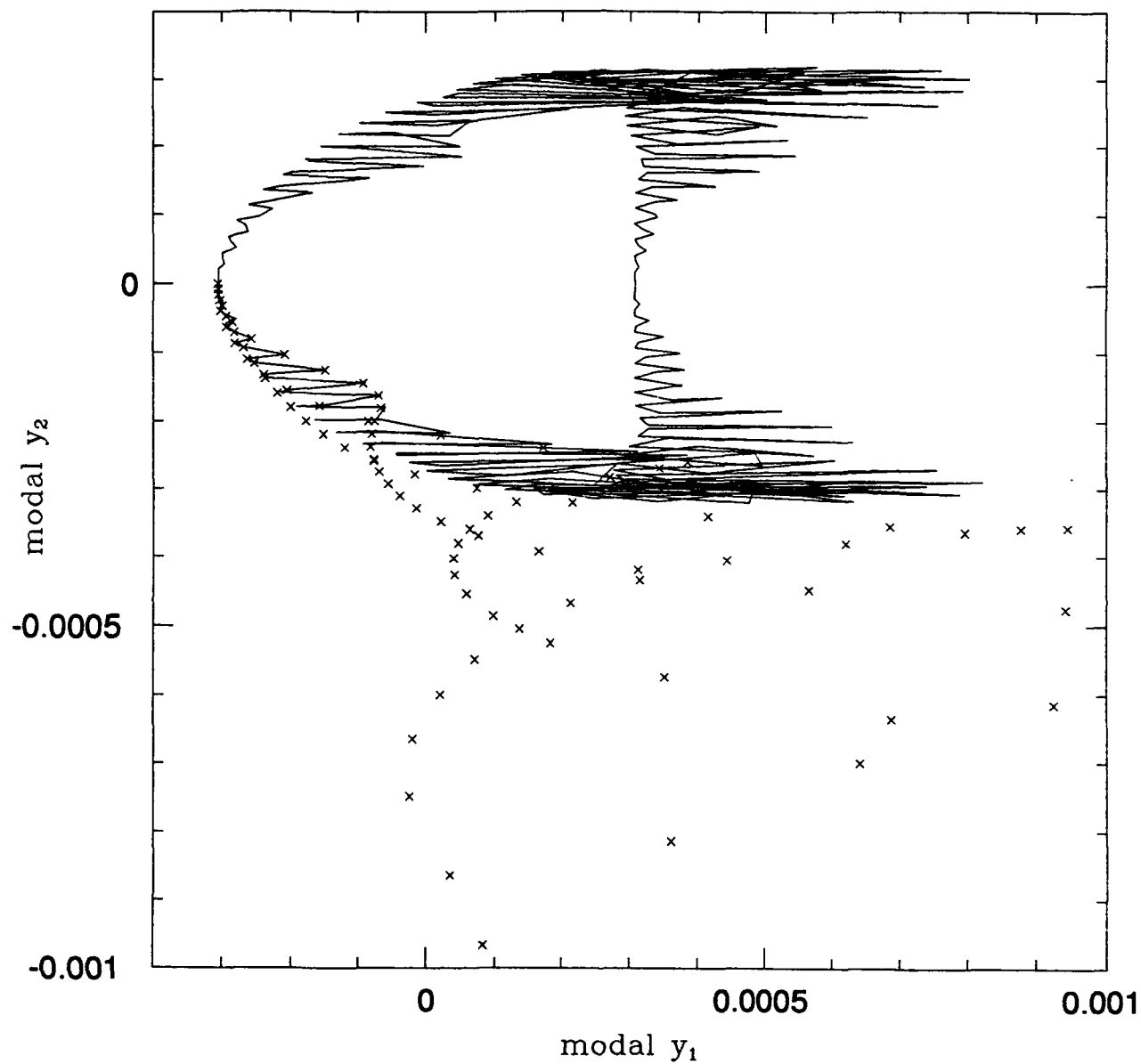


Figure 22: Oscillatory Modal Variables,
 y_1 versus y_2 , $\delta q_1 = 1e-4$

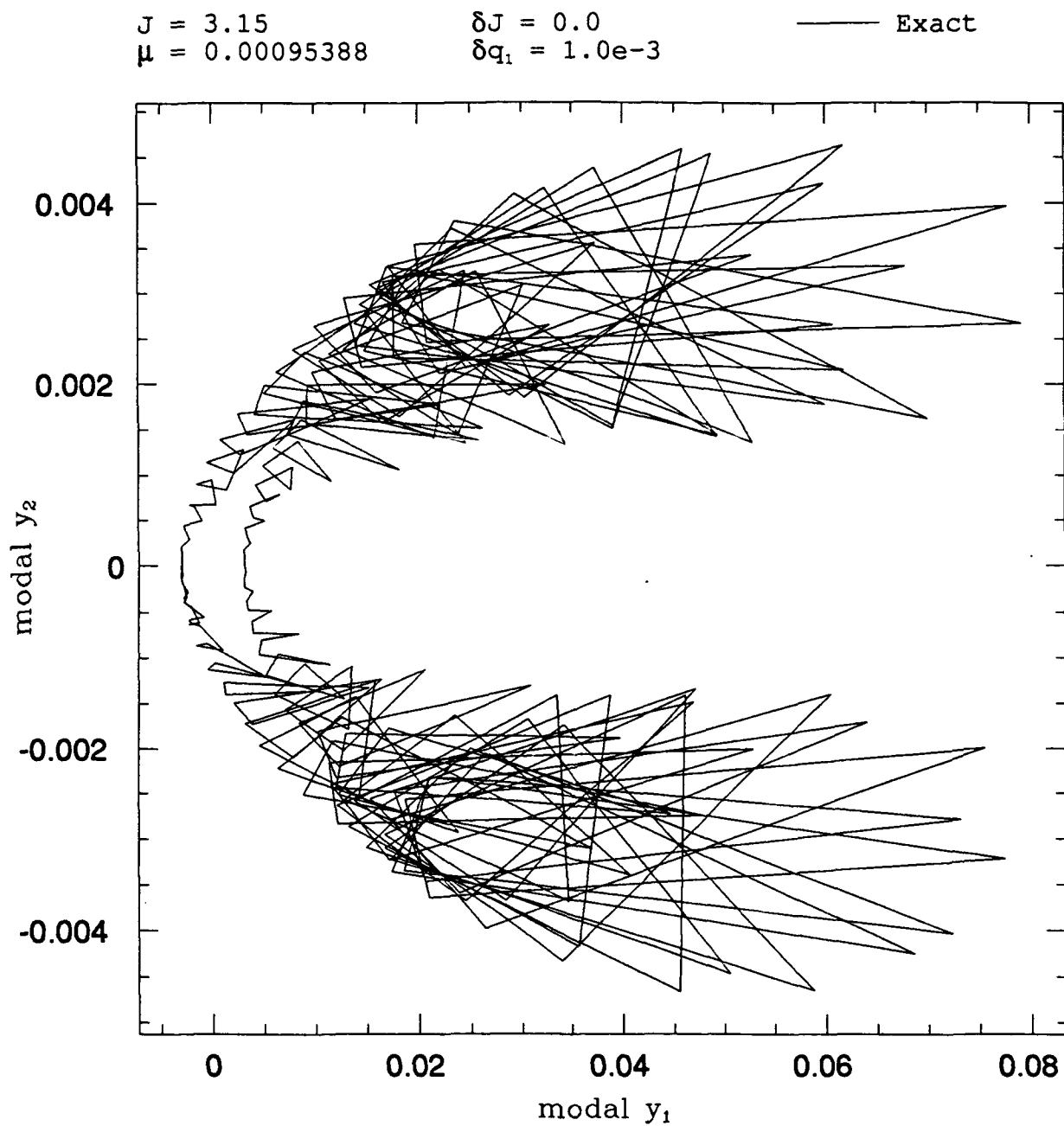


Figure 23: Oscillatory Modal Variables,
 y_1 versus y_2 , Exact Solution, $\delta q_1 = 1e-3$

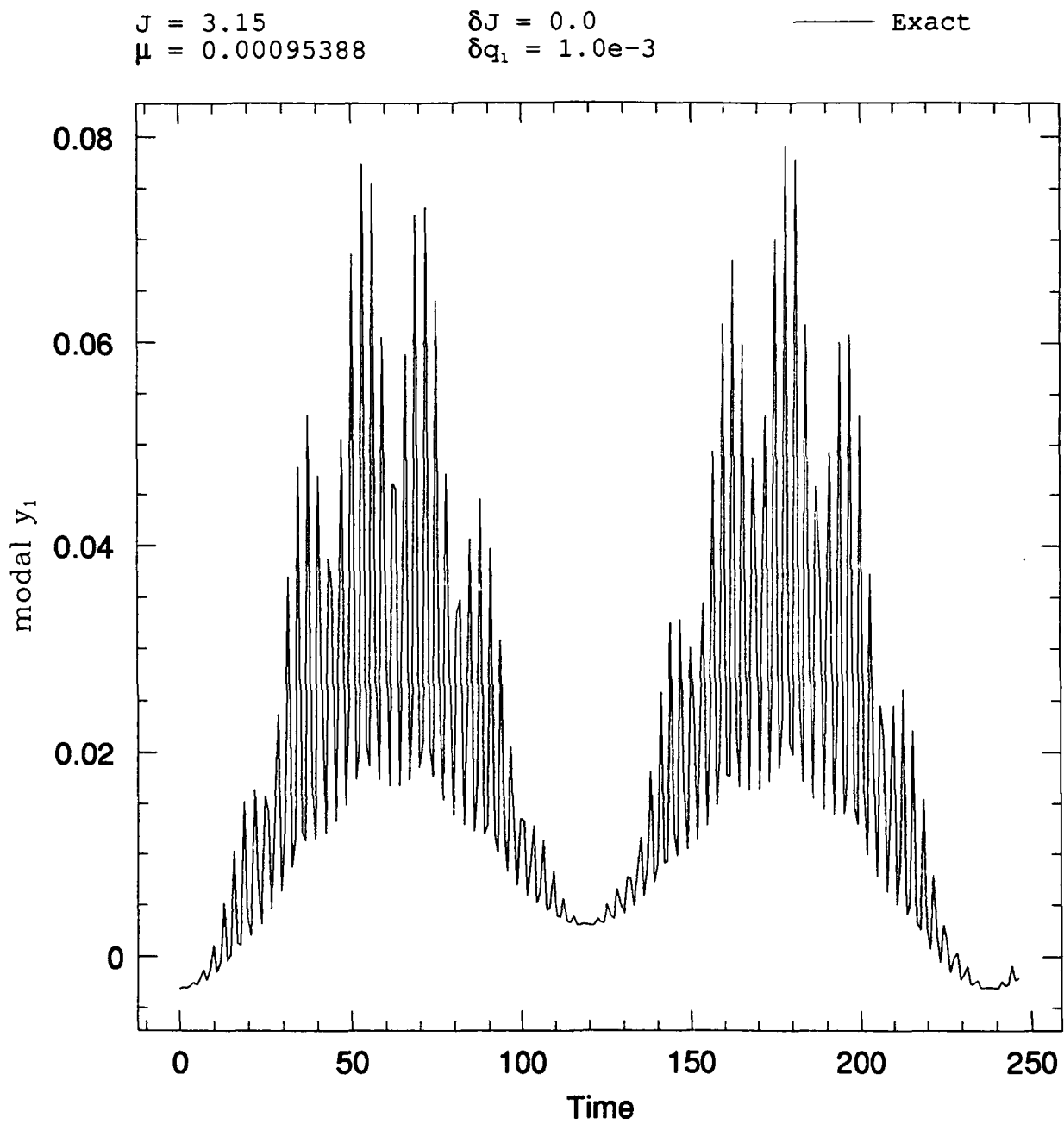


Figure 24: Time History of Modal Variable y_1 , Exact Solution, $\delta q_1 = 1e-3$

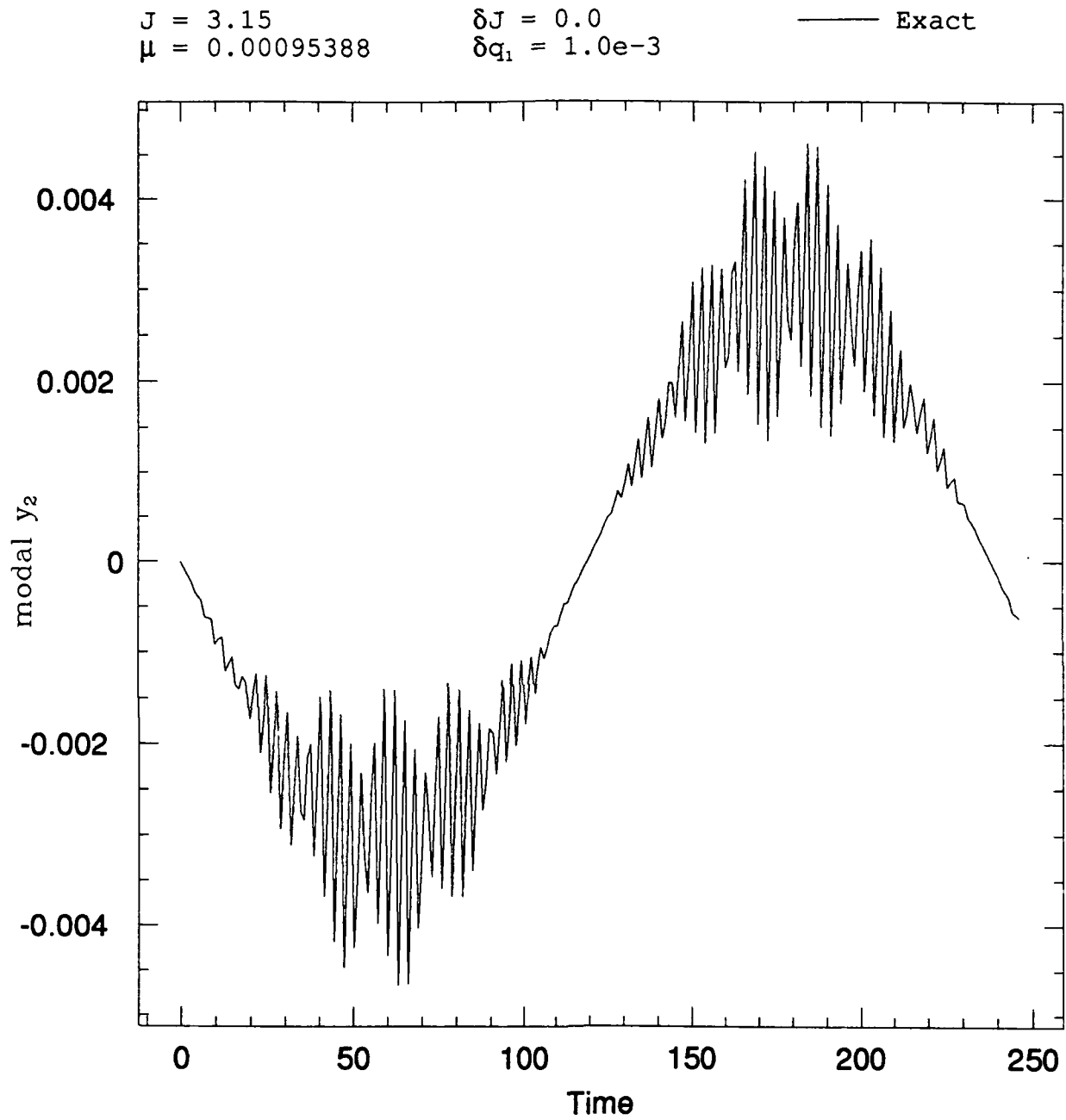


Figure 25: Time History of Modal Variable y_2 , Exact Solution, $\delta q_1 = 1e-3$

and y_4 modals has begun to lose coherence and the value of these modals is above the double precision ability of the computer in both cases. In either solution, the system is no longer periodic.

5.2.2.2 Changes in J

As with the δq_1 cases, the δJ cases will start with $\delta J = 1e-8$. The change in J , or H , is not as easy to conceptualize as the change in q_1 , but the results are just as interesting. Figure 26 shows that the y_1 and y_2 modals again form a visually perfect circle at the initial change in the Jacobian/Hamiltonian and again, the exact and expanded solution match perfectly. The first two modals still describe their sine and cosine waves and are at this point still smooth in structure. Meanwhile, the modals y_3 and y_4 are still too small to draw any significant conclusions. As the change in the Jacobian value is increased to $\delta J = 1e-6$, a similar distortion to the circular plot of y_1 and y_2 begins to appear. This distortion can be seen in Figure 27 for the two modals plotted versus each other and in Figures 28 and 29 for the time history of the modals. Modals y_3 and y_4 are also shown in Figures 30 and 31 for the same change in the Jacobian. At this point, there are two notable differences in the modals with a change in the Jacobian as compared to those with a change in q_1 . First, modals y_1 and y_3 tend to be more erratic in the earlier part of their time history. It is surmised that

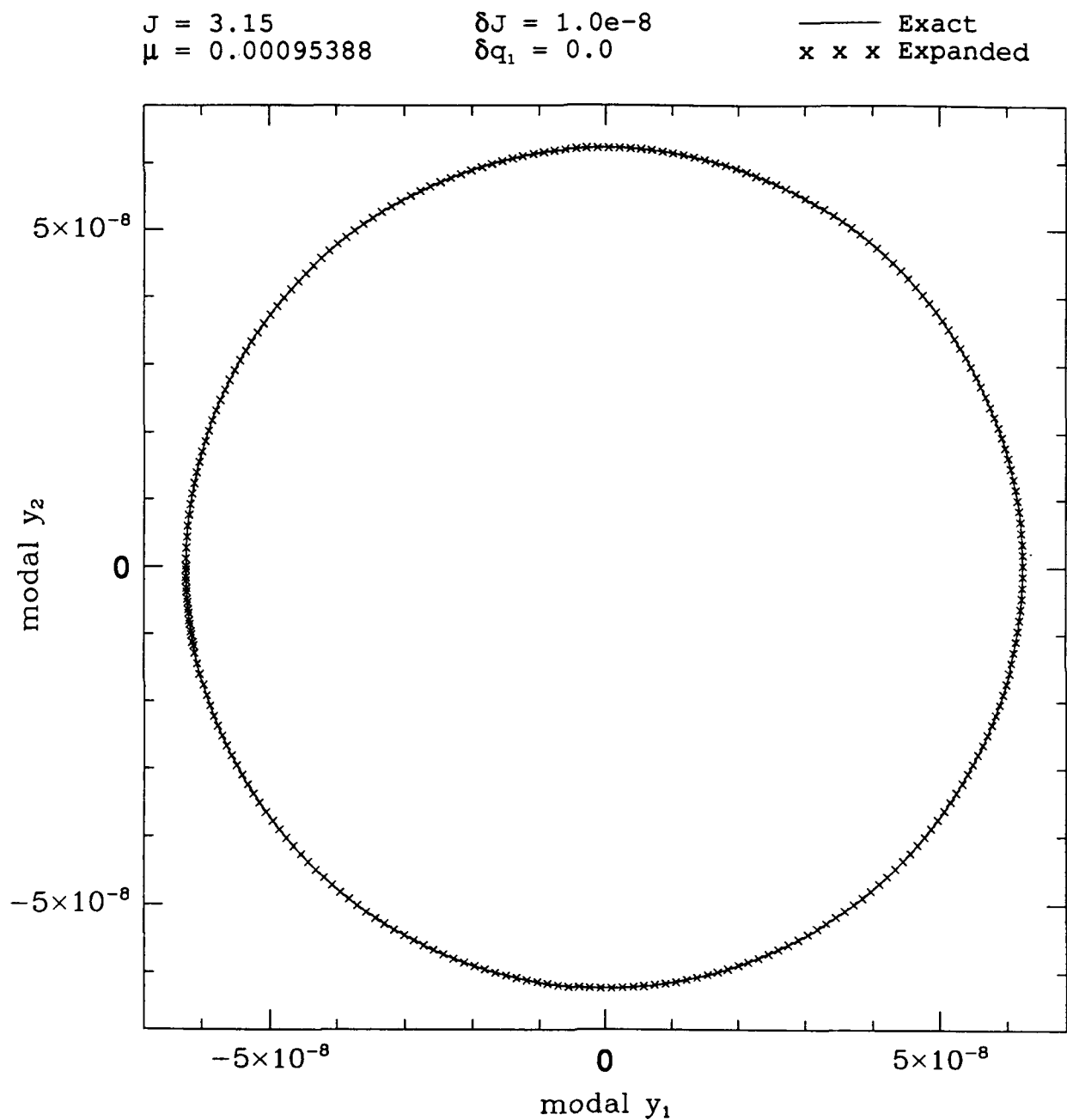


Figure 26: Oscillatory Modal Variables
 y_1 versus y_2 , $\delta J = 1e-8$

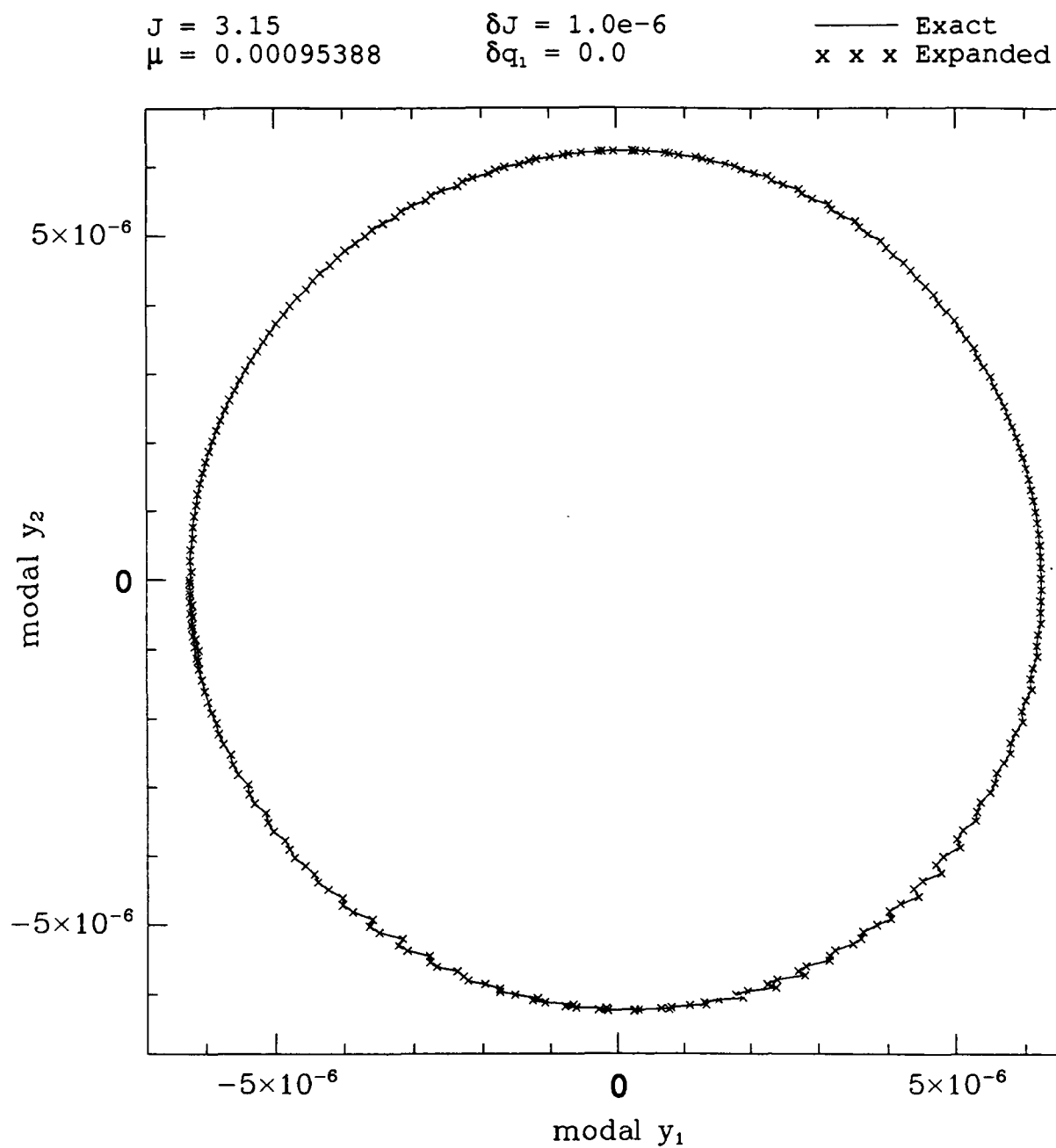


Figure 27: Oscillatory Modal Variables
 y_1 versus y_2 , $\delta J = 1e-6$

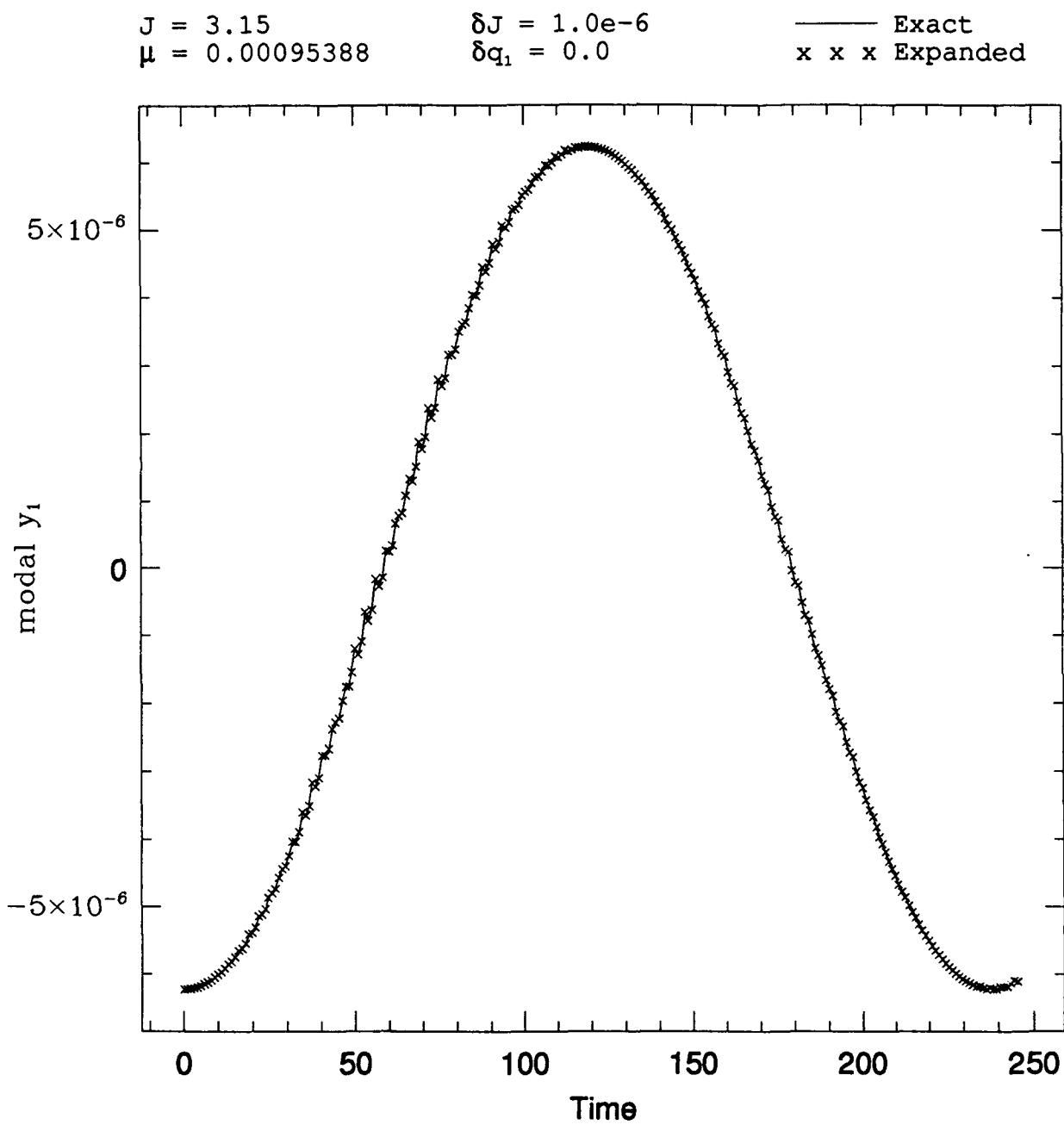


Figure 28: Time History of Modal Variable y_1 , $\delta J = 1e-6$

$J = 3.15$
 $\mu = 0.00095388$

$\delta J = 1.0e-6$
 $\delta q_1 = 0.0$

— Exact
x x x Expanded

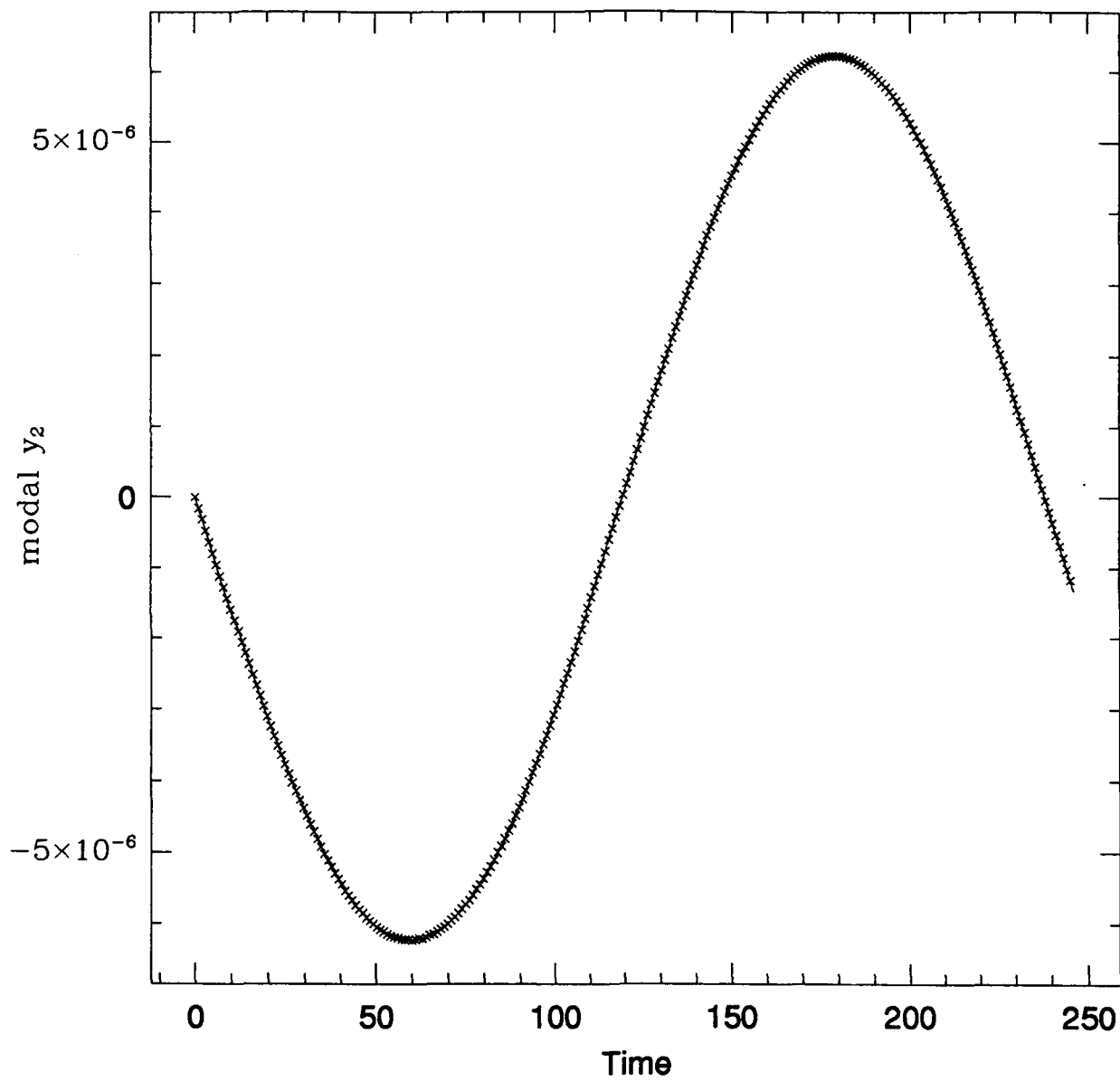


Figure 29: Time History of Modal Variable y_2 , $\delta J = 1e-6$

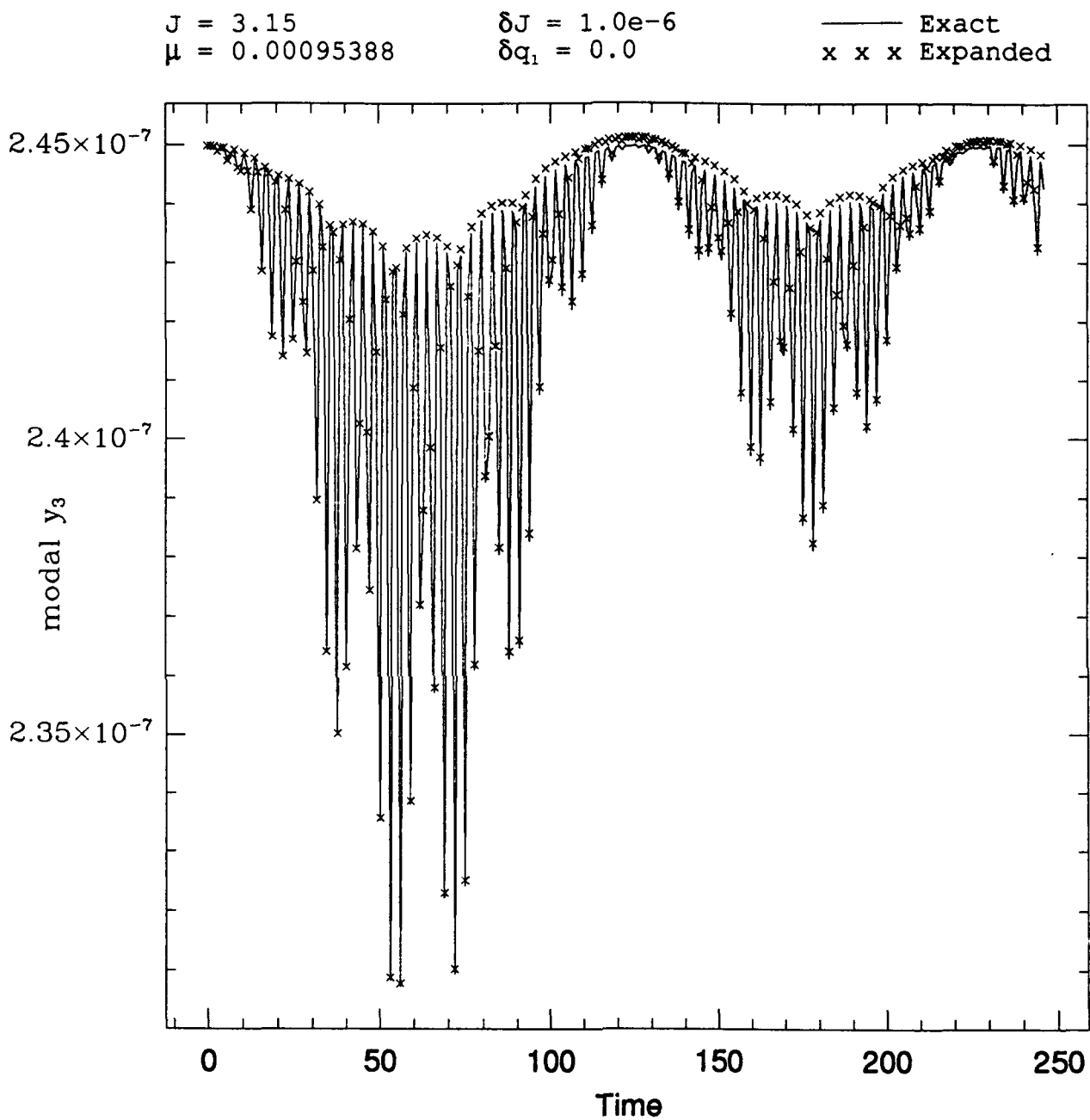


Figure 30: Time History of Modal Variable y_3 , $\delta J = 1e-6$

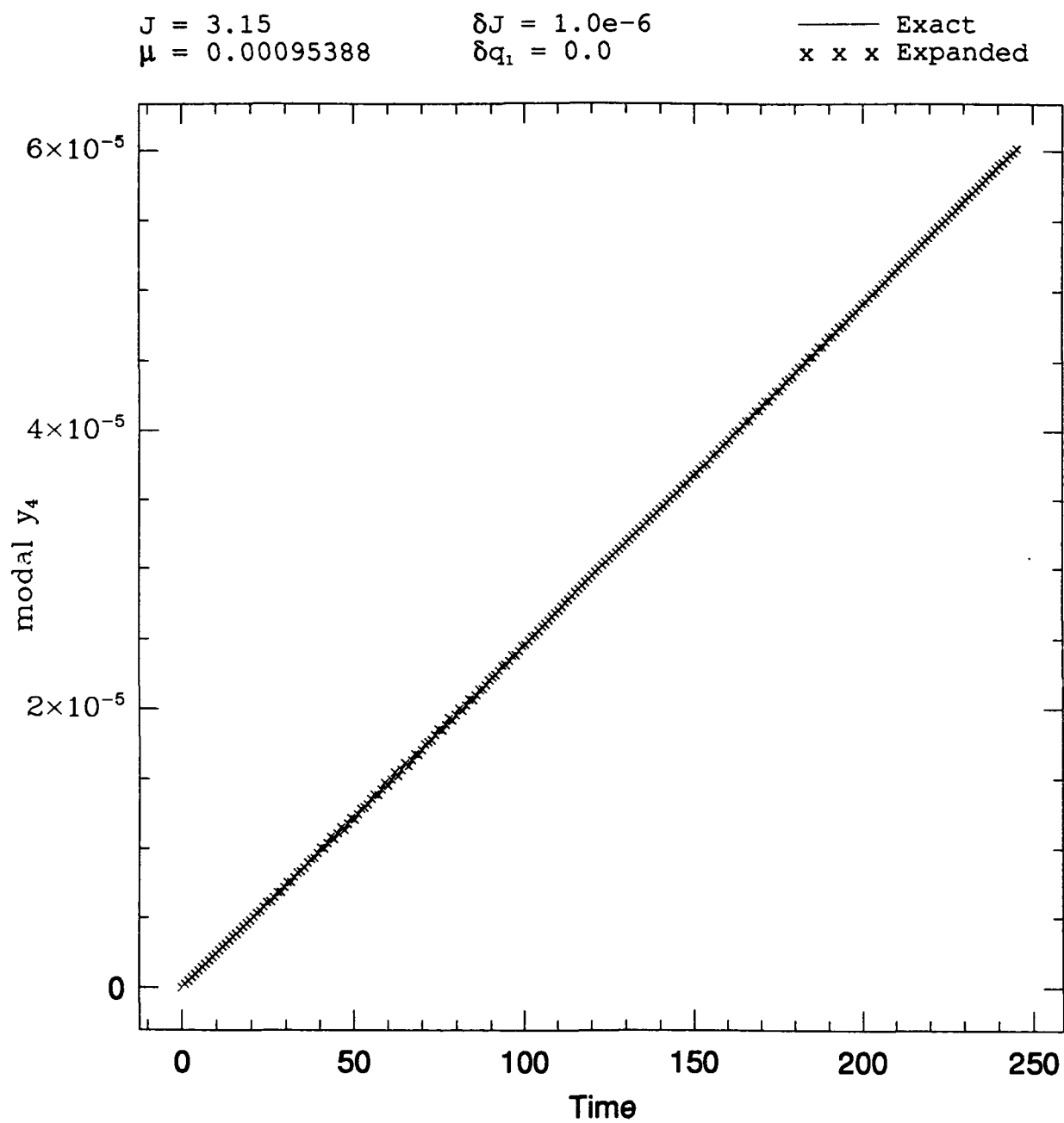


Figure 31: Time History of Modal Variable y_4 , $\delta J = 1e-6$

this increased erratic behavior at the beginning is damped out later due to the influence of the primary masses. The result is a change in the trajectory from a less stable to a more stable state for the new value of the Jacobian/Hamiltonian. Secondly, the y_4 modal continually increases in magnitude throughout the integration period; in fact, it is exactly a linear increase to the modal value. The most likely cause of this is that this modal condition is associated with the time along the trajectory, and the change in the Hamiltonian value causes a constant drift in the time location.

The change in the Jacobian is then increased to $\delta J = 1e-5$ and $\delta J = 5e-5$, and several new deviations are observed. In Figures 32 and 33, the modals y_1 versus y_2 , are notably not converging on each other at the initial condition (located at nine o'clock on the plot), unlike the case in all the y_1 versus y_2 plots for the δq_1 cases. Yet, like the δq_1 cases, there are still two "calm" regions in the path of the modal circles. The first occurs just above the three o'clock point on the path and the second occurs at about ten o'clock. In comparison, looking at Figures 15 and 22, even in the highly perturbed cases these regions appear at exactly three and nine o'clock. A similar pattern is seen when comparing the y_1 modal time histories in Figures 34 and 35 to those of Figures 17 and 24. This would also support the idea that there is a drift along the time domain

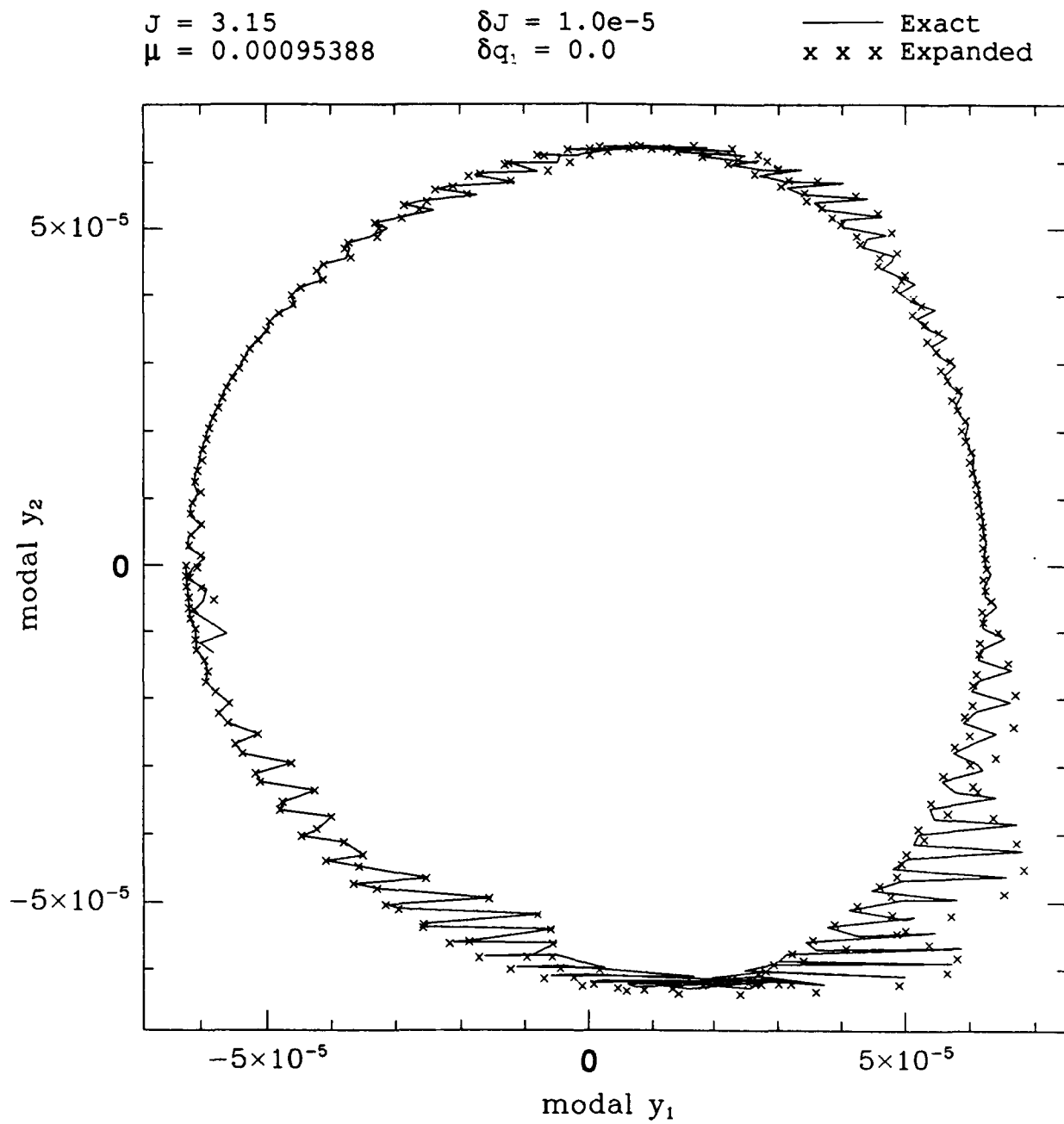


Figure 32: Oscillatory Modal Variables,
 y_1 versus y_2 , $\delta J = 1e-5$

$J = 3.15$
 $\mu = 0.00095388$

$\delta J = 5.0e-5$
 $\delta q_1 = 0.0$

— Exact
 x x x Expanded

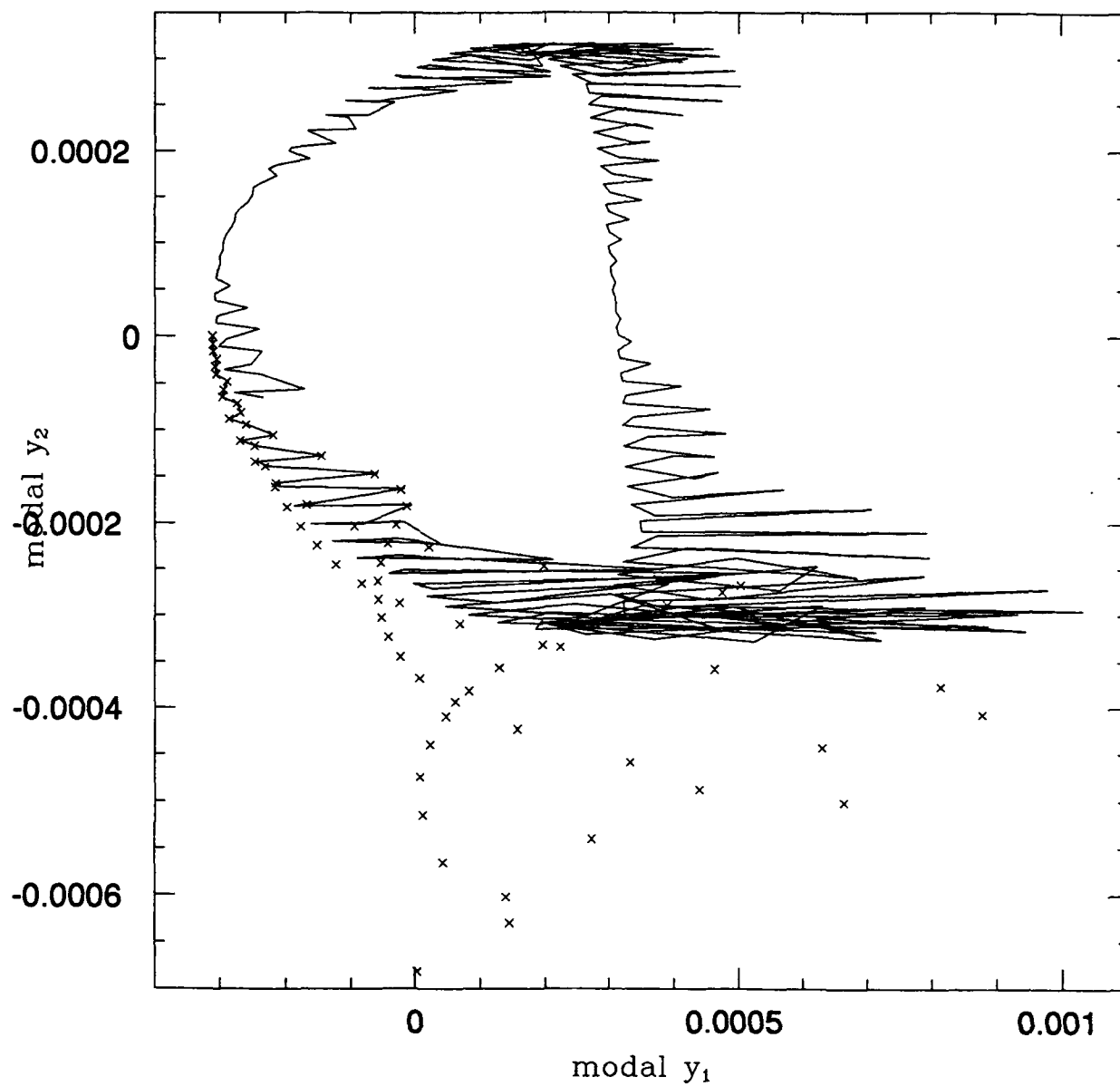


Figure 33: Oscillatory Modal Variables,
 y_1 versus y_2 , $\delta J = 5e-5$

$J = 3.15$
 $\mu = 0.00095388$

$\delta J = 1.0e-5$
 $\delta q_1 = 0.0$

— Exact
x x x Expanded

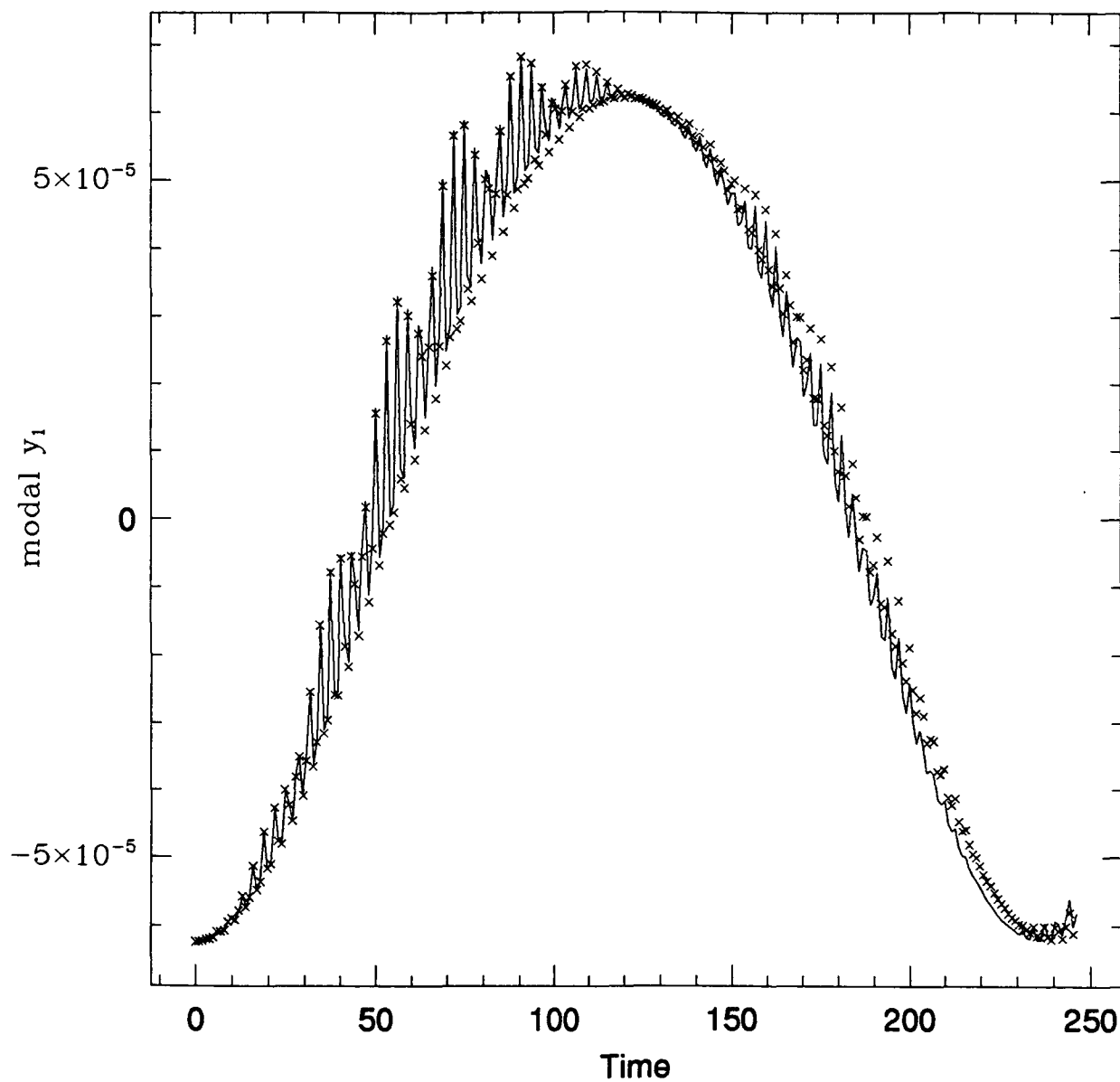


Figure 34: Time History of Modal
Variable y_1 , $\delta J = 1e-5$

$J = 3.15$
 $\mu = 0.00095388$

$\delta J = 5.0e-5$
 $\delta q_1 = 0.0$

— Exact

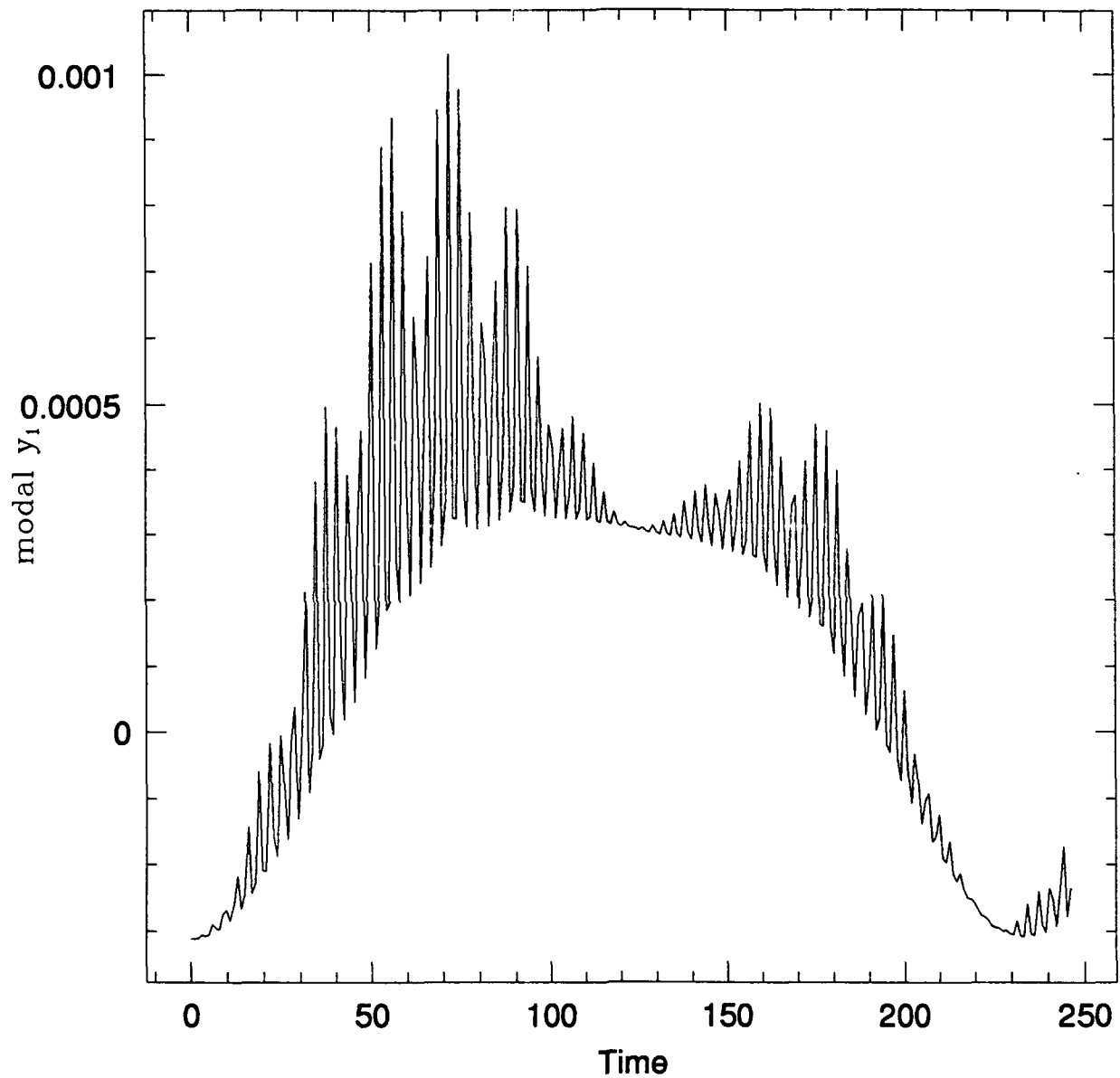


Figure 35: Time History of Modal Variable y_1 , Exact Solution, $\delta J = 5e-5$

of the system and that it is affecting all the modal variables. Also note, that through $\delta J = 1e-5$, the expanded solution has closely followed the exact solution. But, at $\delta J = 5e-5$, the expanded solution again falls off.

Finally, the change in the Jacobian/Hamiltonian is increased to $\delta J = 1e-3$, where the same trends noted before are amplified in magnitude. A close look at Figures 36 through 40 shows that the second erratic region for each modal variable is shorter and smaller in magnitude than the first, and it appears that a third region is starting. If these regions continue to get shorter and smaller, it would support the theory that the perturbed trajectory is settling into a new trajectory that may be more stable than the original.

5.3 *Summary*

In the end analysis, it appears that the exact solution tracks the modal variables slightly better at larger perturbations than the expanded solution. Since the correlation of the two solutions is so accurate at small perturbations, this suggests that the biggest fault in the expanded solution is the truncation of the Hamiltonian after the cubic term. The nearly perfect comparison of the two solutions also lends credibility to both methods used in this study for the perturbation problem. Because the perturbation problem was secondary to this study, the interpretation of the perturbed solutions presented may not

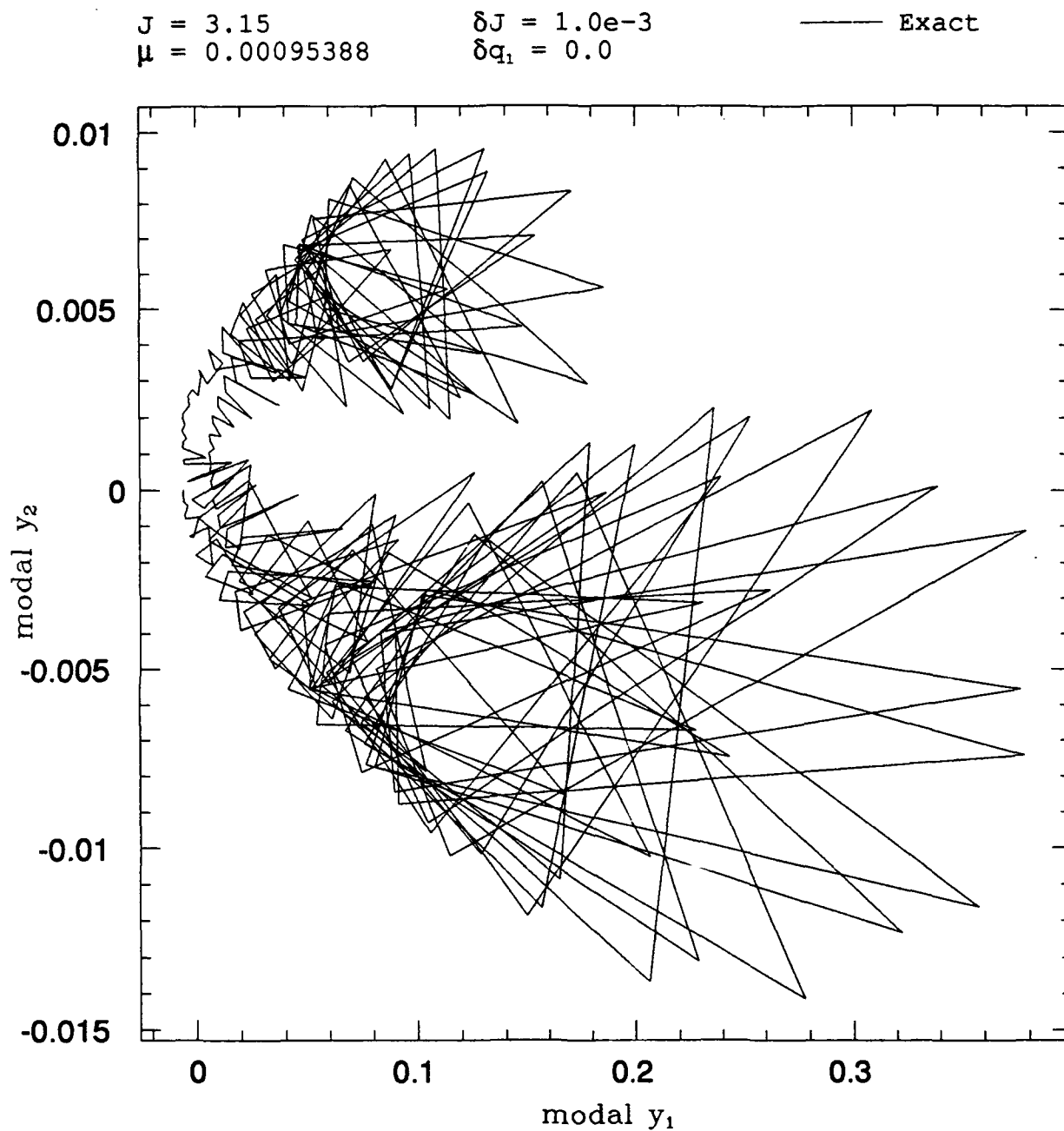


Figure 36: Oscillatory Modal Variables,
 y_1 versus y_2 , Exact Solution, $\delta J = 1e-3$

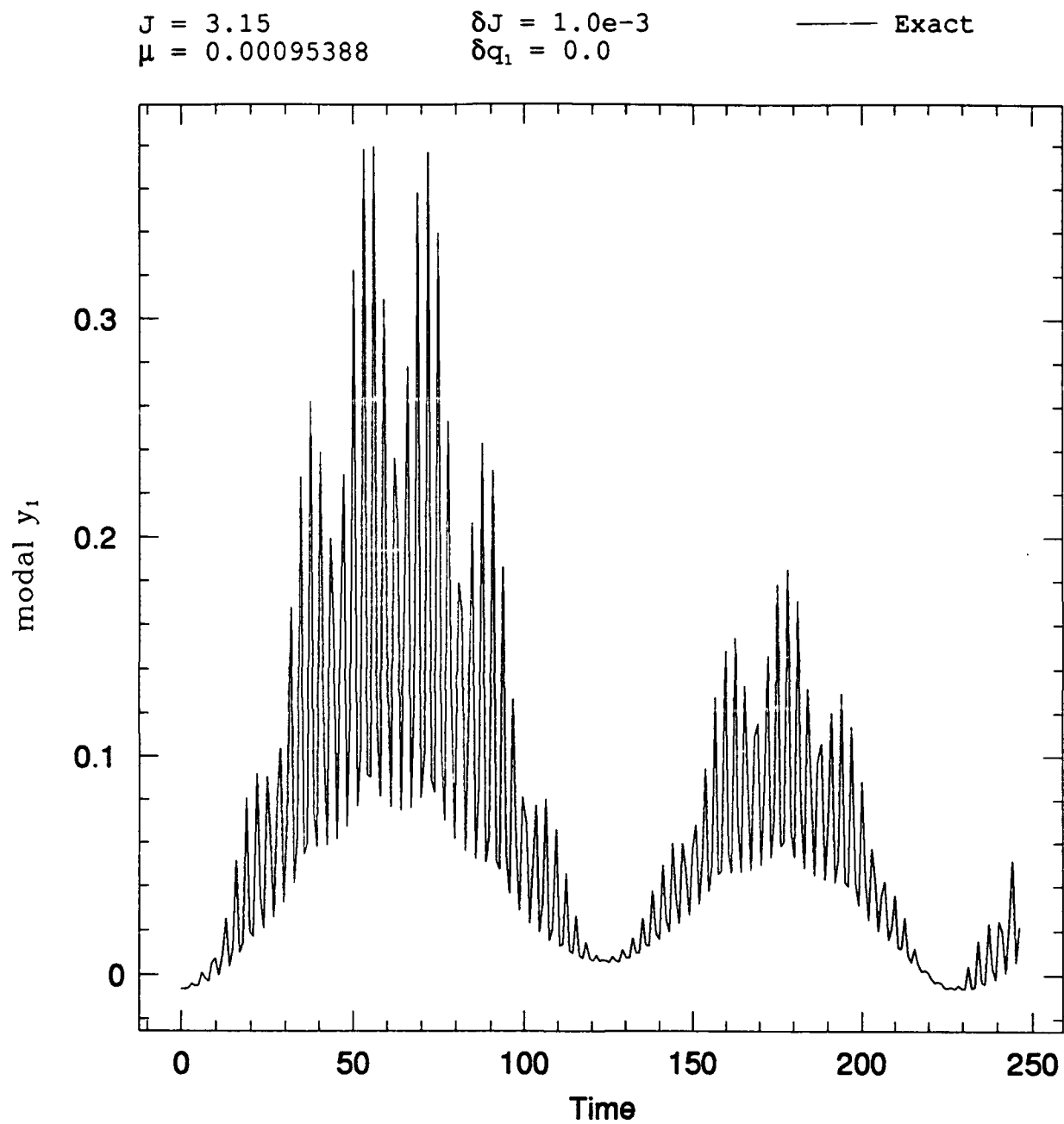


Figure 37: Time History of Modal Variable y_1 , Exact Solution, $\delta J = 1e-3$

$J = 3.15$
 $\mu = 0.00095388$

$\delta J = 1.0e-3$
 $\delta q_1 = 0.0$

— Exact

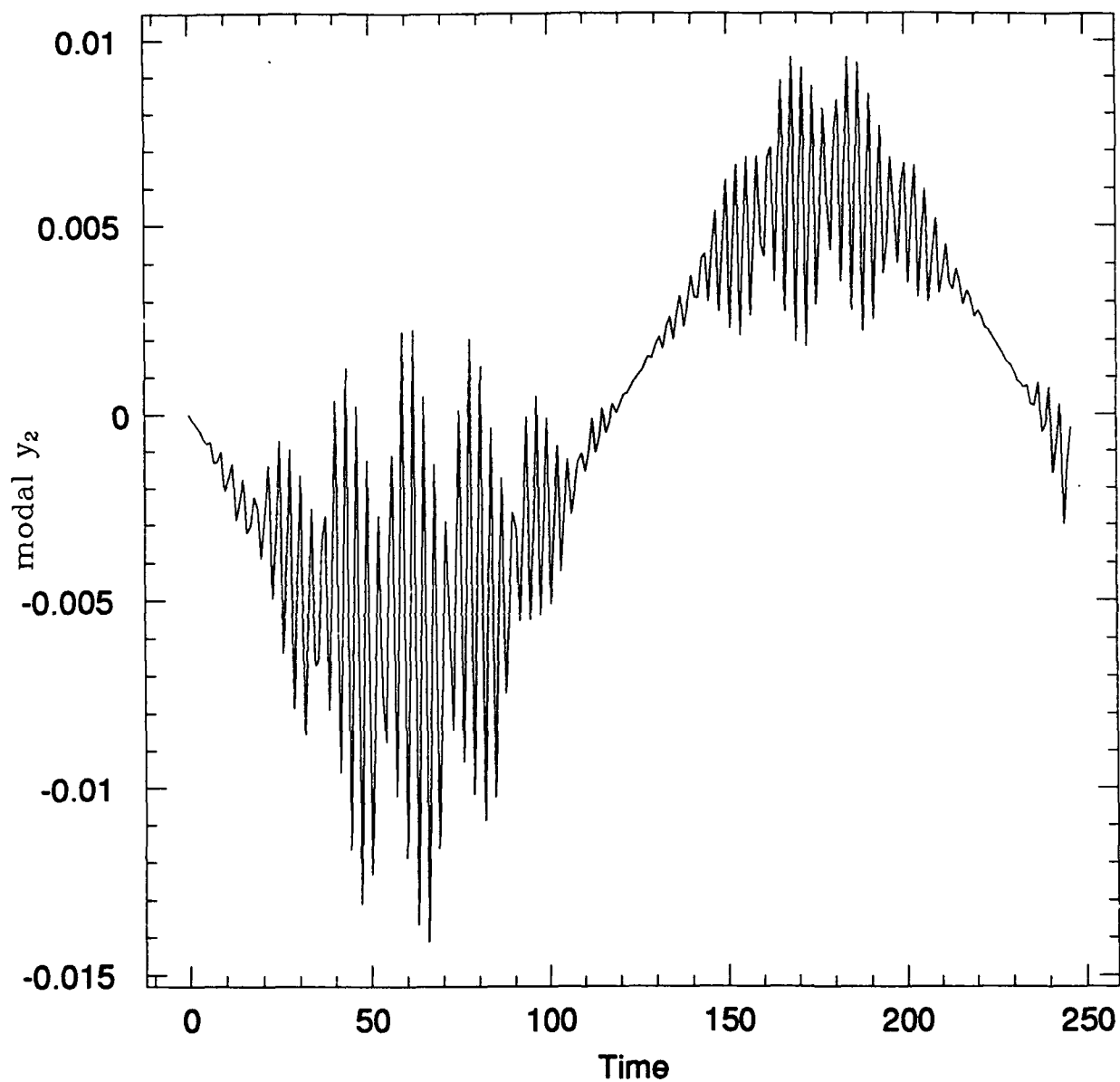


Figure 38: Time History of Modal Variable y_2 , Exact Solution, $\delta J = 1e-3$

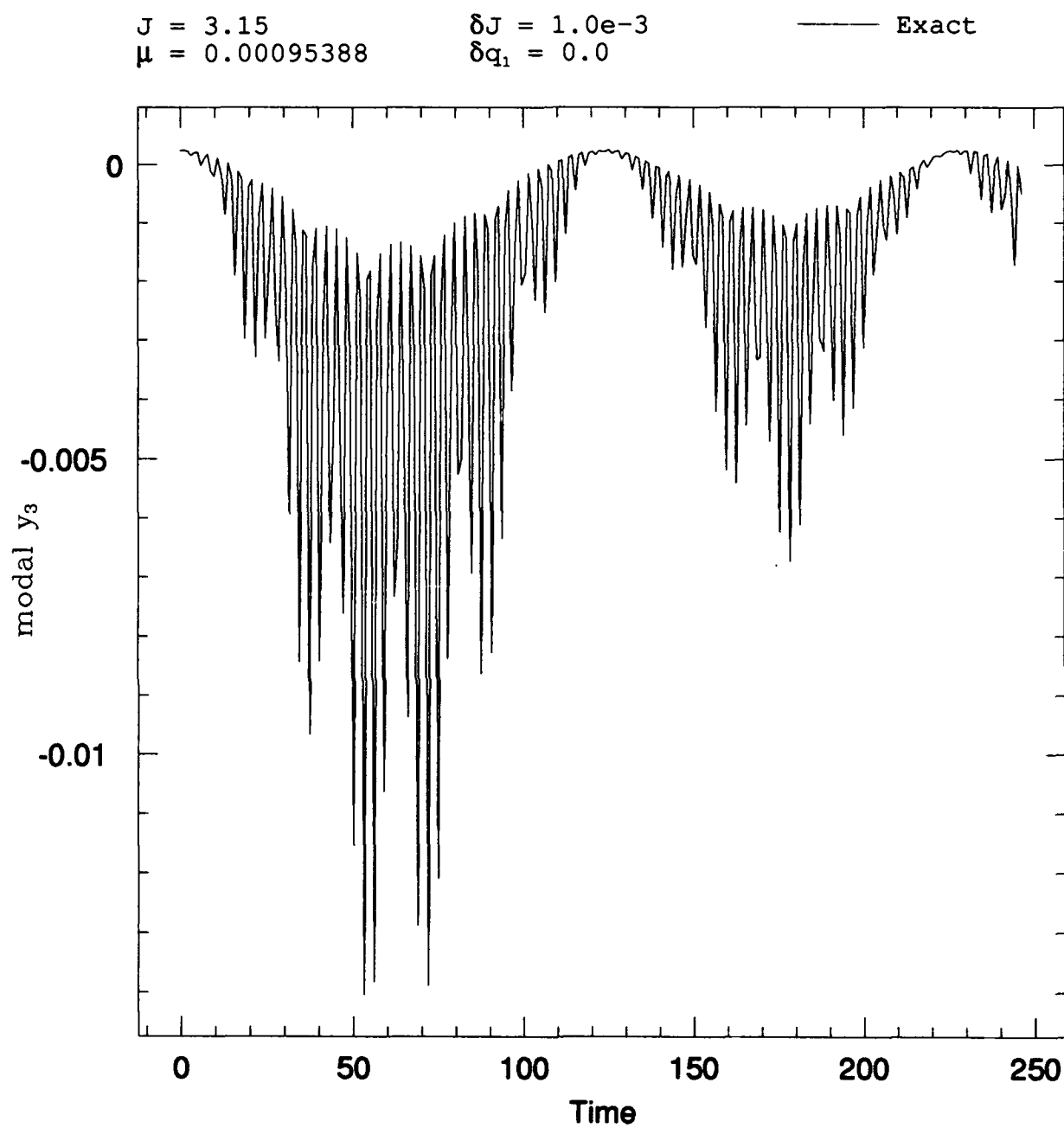


Figure 39: Time History of Modal Variable y_3 , Exact Solution, $\delta J = 1e-3$

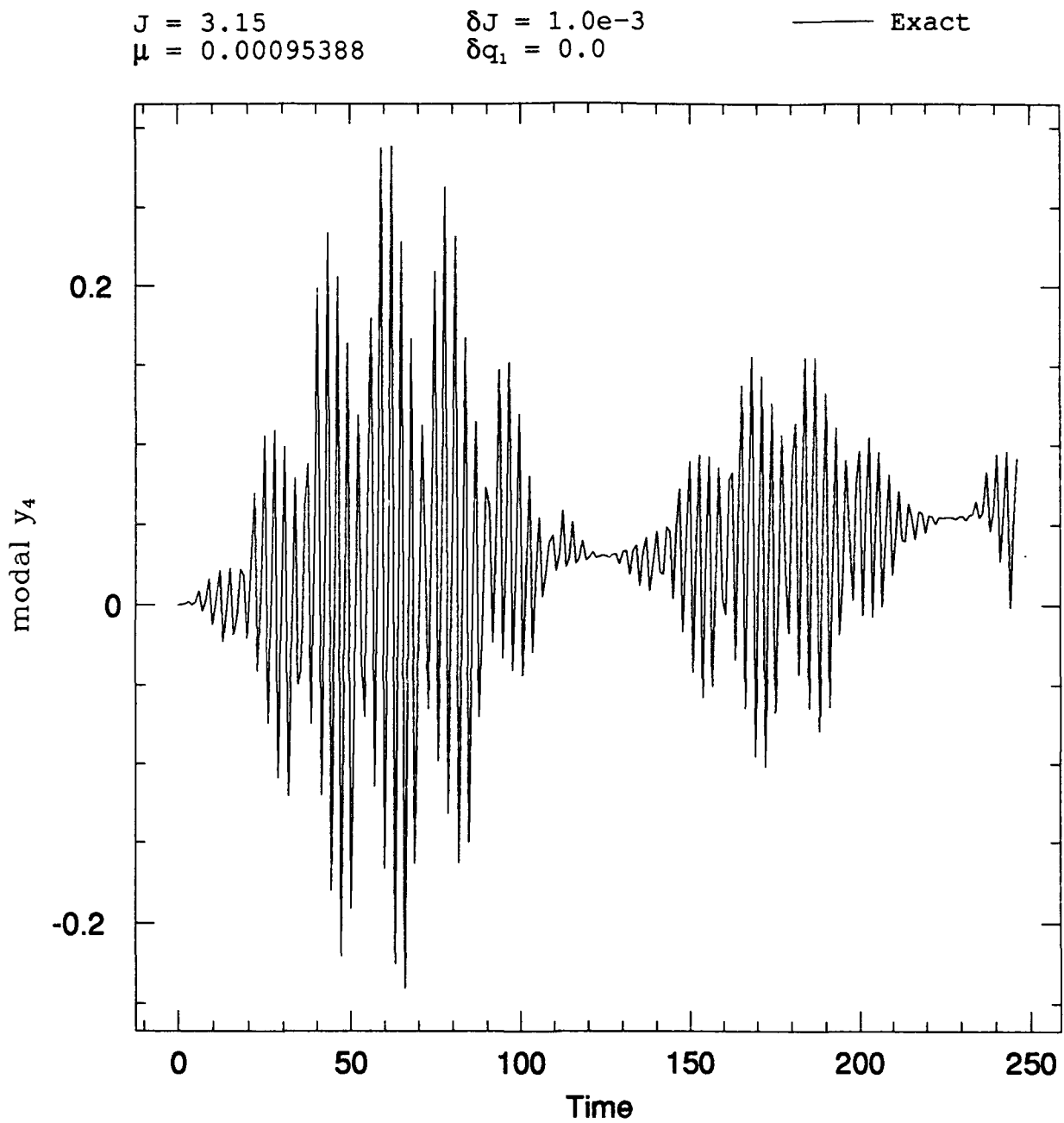


Figure 40: Time History of Modal Variable y_4 , Exact Solution, $\delta J = 1e-3$

be the only explanation and the reader is encouraged to make their own examination of this and other periodic solutions using the canonical Floquet theory.

VI. Conclusions and Recommendations

The major conclusion of this study is that Floquet theory can be used for canonical transformations to real-valued modal variables for use in perturbation research on periodic systems. Any transformation, $\mathbf{T}$, to the eigenvectors or exponents of the Floquet solution must obey the relation $\mathbf{T}^T \mathbf{Z} \mathbf{T} = \mathbf{Z}$ in order for the transformation to be a canonical one. Special considerations must be made for each Poincaré exponent pair type, with extra emphasis on the degenerate case of zero Poincaré exponents.

In support of the major finding, the restricted three-body solution was analyzed with both an exact and an expanded solution in the modal variables. The results showed that the exact and expanded solutions agreed extremely well for very small perturbation. At larger perturbations, as expected, the expanded solution was no longer accurate due to the truncation of the expansion after the cubic term. The exact solution, while remaining graphically periodic at higher perturbations, also exhibited irregular displacements at these higher perturbations and would likely result in complete loss of the periodic reference trajectory over time.

The recommended follow-on to this work would involve a much more detailed analysis of the three-body problem to include expanding the analysis to a three-dimensional

problem. Mass systems other than the Sun-Jupiter system would create more interesting surface of section plots and would doubtlessly result in much more interesting modal analyses. There is also the need for using canonical Floquet theory on other periodic systems, not only to further validate this approach to canonical transformations, but to assess the full usefulness of this approach in perturbation problems. Another system that would be of particular interest, because of its requirement for precise operation, would be that of rotating blades on helicopters and jet engines.

Appendix A: Six Modal Equations of Variation Types

All six cases for the modal equations of variation are listed in the following pages. Note that the $\partial \mathcal{H} / \partial y_1$ is equivalent to $\partial (K_2) / \partial y_1$ term for the expansion of the variational modal Hamiltonian.

Case 1: Poincaré exponents are a positive/negative pair of reals, and the symplectic eigenvectors are real. The initial and final forms of $\mathbf{J}$ are

$$\mathbf{J} = \begin{bmatrix} \omega & 0 \\ 0 & -\omega \end{bmatrix} \quad (\text{A-1})$$

the resulting $\mathbf{S}$ is then

$$\mathbf{S} = -\mathbf{ZJ} = \begin{bmatrix} 0 & \omega \\ \omega & 0 \end{bmatrix} \quad (\text{A-2})$$

and the modal Hamiltonian $\mathfrak{K}$ is

$$\mathfrak{K} = \frac{1}{2} \bar{\mathbf{y}}^T \mathbf{S} \bar{\mathbf{y}} = \frac{1}{2} (y_1 y_2 \omega + y_1 y_2 \omega) = y_1 y_2 \omega \quad (\text{A-3})$$

Finally, the equations of variation are

$$\begin{aligned} \dot{y}_1 &= \frac{\partial \mathfrak{K}}{\partial y_2} = y_1 \omega \\ \dot{y}_2 &= -\frac{\partial \mathfrak{K}}{\partial y_1} = -y_2 \omega \end{aligned} \quad (\text{A-4})$$

Case 2: Poincaré exponents are a positive/negative pair of reals, and the symplectic eigenvectors are imaginary. The forms of $\mathbf{J}$ are

$$\mathbf{J} = \begin{bmatrix} \omega & 0 \\ 0 & -\omega \end{bmatrix} \quad \mathbf{J}'' = \begin{bmatrix} -\omega & 0 \\ 0 & \omega \end{bmatrix} \quad (\text{A-5})$$

and the resulting $\mathbf{S}''$ is

$$\mathbf{S}'' = -\mathbf{Z}\mathbf{J}'' = \begin{bmatrix} 0 & -\omega \\ -\omega & 0 \end{bmatrix} \quad (\text{A-6})$$

and the modal Hamiltonian $\mathfrak{K}$ is

$$\mathfrak{K} = \frac{1}{2} \bar{\mathbf{y}}^T \mathbf{S}'' \bar{\mathbf{y}} = \frac{1}{2} (-y_1 y_2 \omega - y_1 y_2 \omega) = -y_1 y_2 \omega \quad (\text{A-7})$$

Finally, the equations of variation are

$$\begin{aligned} \dot{y}_1 &= \frac{\partial \mathfrak{K}}{\partial y_2} = -y_1 \omega \\ \dot{y}_2 &= -\frac{\partial \mathfrak{K}}{\partial y_1} = y_2 \omega \end{aligned} \quad (\text{A-8})$$

Case 3: Poincaré exponents are a positive/negative pair of imaginaries, and the symplectic eigenvectors are real. The forms of $\mathbf{J}$ are

$$\mathbf{J} = \begin{bmatrix} \omega t & 0 \\ 0 & -\omega t \end{bmatrix} \quad \mathbf{J}' = \begin{bmatrix} 0 & -\omega \\ \omega & 0 \end{bmatrix} \quad (\text{A-9})$$

and the resulting $\mathbf{S}'$ is

$$\mathbf{S}' = -\mathbf{Z}\mathbf{J}' = \begin{bmatrix} -\omega & 0 \\ 0 & -\omega \end{bmatrix} \quad (\text{A-10})$$

and the modal Hamiltonian $\mathfrak{H}$ is

$$\mathfrak{H} = \frac{1}{2} \bar{\mathbf{y}}^T \mathbf{S}' \bar{\mathbf{y}} = -\frac{1}{2} \omega (y_1^2 + y_2^2) \quad (\text{A-11})$$

Finally, the equations of variation are

$$\begin{aligned} \dot{y}_1 &= \frac{\partial \mathfrak{H}}{\partial y_2} = -y_2 \omega \\ \dot{y}_2 &= -\frac{\partial \mathfrak{H}}{\partial y_1} = y_1 \omega \end{aligned} \quad (\text{A-12})$$

Case 4: Poincaré exponents are a positive/negative pair of imaginaries, and the symplectic eigenvectors are imaginary. The forms of $\mathbf{J}$ are

$$\mathbf{J} = \begin{bmatrix} \omega t & 0 \\ 0 & -\omega t \end{bmatrix} \quad \mathbf{J}' = \begin{bmatrix} 0 & -\omega \\ \omega & 0 \end{bmatrix} \quad \mathbf{J}'' = \begin{bmatrix} 0 & \omega \\ -\omega & 0 \end{bmatrix} \quad (\text{A-13})$$

and the resulting $\mathbf{S}''$ is

$$\mathbf{S}'' = -\mathbf{Z}\mathbf{J}'' = \begin{bmatrix} \omega & 0 \\ 0 & \omega \end{bmatrix} \quad (\text{A-14})$$

and the modal Hamiltonian $\mathfrak{K}$ is

$$\mathfrak{K} = \frac{1}{2} \bar{\mathbf{y}}^T \mathbf{S}'' \bar{\mathbf{y}} = \frac{1}{2} \omega (y_1^2 + y_2^2) \quad (\text{A-15})$$

Finally, the equations of variation are

$$\begin{aligned} \dot{y}_1 &= \frac{\partial \mathfrak{K}}{\partial y_2} = y_2 \omega \\ \dot{y}_2 &= -\frac{\partial \mathfrak{K}}{\partial y_1} = -y_1 \omega \end{aligned} \quad (\text{A-16})$$

Case 5: Poincaré exponents are a pair of zeros, the degenerate case, and the symplectic eigenvectors are real. The initial and final form of $\mathbf{J}$ are

$$\mathbf{J} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \quad (\text{A-17})$$

and the resulting $\mathbf{S}$ is

$$\mathbf{S} = -\mathbf{ZJ} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \quad (\text{A-18})$$

and the modal Hamiltonian $\mathfrak{K}$ is

$$\mathfrak{K} = \frac{1}{2} \bar{\mathbf{y}}^T \mathbf{S} \bar{\mathbf{y}} = \frac{1}{2} y_2^2 \quad (\text{A-19})$$

Finally, the equations of variation are

$$\begin{aligned} \dot{y}_1 &= \frac{\partial \mathfrak{K}}{\partial y_2} = y_2 \\ \dot{y}_2 &= -\frac{\partial \mathfrak{K}}{\partial y_1} = 0 \end{aligned} \quad (\text{A-20})$$

Case 6: Poincaré exponents are a pair of zeros, the degenerate case, and the symplectic eigenvectors are imaginary. The forms of $\mathbf{J}$ are

$$\mathbf{J} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \quad \mathbf{J}'' = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \quad (\text{A-21})$$

and the resulting $\mathbf{S}''$ is

$$\mathbf{S}'' = -\mathbf{z}\mathbf{J}'' = \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix} \quad (\text{A-22})$$

and the modal Hamiltonian $\mathfrak{K}$ is

$$\mathfrak{K} = \frac{1}{2}\bar{\mathbf{y}}^T \mathbf{S}'' \bar{\mathbf{y}} = -\frac{1}{2}y_1^2 \quad (\text{A-23})$$

Finally, the equations of variation are

$$\begin{aligned} \dot{y}_1 &= \frac{\partial \mathfrak{K}}{\partial y_2} = 0 \\ \dot{y}_2 &= -\frac{\partial \mathfrak{K}}{\partial y_1} = y_1 \end{aligned} \quad (\text{A-24})$$

Appendix B: Fortran Code

```

C      program section
C
C      ***** "SURFACE OF SECTION" *****
C      - creates surface of section plot for the restricted
C      three body perturbation problem
C
C      implicit double precision (a-h)
C      implicit integer (i-n)
C      implicit double precision (o-z)
C
C      ***** PROGRAM COMMONS *****
C
C      common /data/ xmu,xmua
C      common /ham/ t,x(20,4),f(20,4),err(20),hh,nn,mode
C
C      dimension rdotv(4),x(20,4),f(20,4),err(20)
C
C      character*10 filename1,filename2,filename3
C
C      ***** READ INPUT DATA *****
C
C      read(*,*) nic
C
C      nic = number of initial conditions
C
C      read(*,*) xmu
C      xmua = 1.d0 - xmu
C      read(*,*) hh,tmax
C      npts = dint(tmax/hh)
C      read(*,*) xjac,syn
C
C      ***** OPEN OUTPUT FILES *****
C      - file 1 is general output
C      - file 2 is input for program period
C      - file 3 is the surface of section plotfile
C
C      read(*,*) filename1
C      open(1,FILE=filename1,STATUS='UNKNOWN')
C      read(*,*) filename2
C      open(2,FILE=filename2,STATUS='UNKNOWN')
C      read(*,*) filename3
C      open(3,FILE=filename3,STATUS='UNKNOWN')
C
C      ***** REPEAT INPUT VALUES TO DATA FILE *****
C
C      write (1,*) 'mu=',xmu,' 1-mu=',xmua
C      write (1,*) 'step=',hh,' maximum time=',tmax
C      write (1,*) 'number of points=',npts
C      write (1,*) 'Jacobian constant=',xjac
C
C      do 600 j = 1,nic
C        k=0
C        read(*,*) xnot,ynot
C        write (1,*)
C        write (1,*) 'x0=',xnot,' y0=',ynot
C
C        mode = 0
C        nn = 4
C        nxt = 0
C        t = 0.d0
C
C      *****
C      GET q1,p1,q2,p2,AND HAMILTONIAN; FOR GIVEN x0,y0, AND JACOBIAN
C      *****
C
C      q1 = xnot+xmu
C      q2 = ynot
C      xham = (xmu*xmua-xjac)/2.d0
C      r1 = dsqrt((q1-xmu)*(q1-xmu) + q2*q2)
C      r2 = dsqrt((q1+xmua)*(q1+xmua) + q2*q2)

```

```

      d = xham + xmu*a/r1 + xmu/r2
      g = q2/(q1-xmu)
      a = 0.5d0*(g*g + 1.d0)
      b = -(g*g*xmu + g*q2 + q1)
      c = 0.5d0*g*g*xmu*xmu + g*q2*xmu - d
      disc = b*b - 4.d0*a*c
      if( disc .lt. 0.d0) then
        write (1,*) 'stop 10 - no roots'
        go to 600
      endif
      p2 = (-b+syn*dsqrt(disc))/(2.d0*a)
      p1 = g*(xmu-p2)
C
C ***** INITIAL CONDITIONS *****
C
      x(1,1) = q1
      x(2,1) = p1
      x(3,1) = q2
      x(4,1) = p2
C
C ***** INITIALIZE HAMING *****
C
      call haming(nxt)
C
C ***** TURN OFF SECOND EOM EVALUATION *****
C
      nxt = -nxt
      if(nxt .ne. 0) go to 499
      write (1,*) 'stop 99 - unable to start Haming'
      go to 600
499      continue
C
C ***** INTEGRATION LOOP *****
C
      write (1,*)
      write (1,*) 'npts value at n*period'
      do 500 i = 1,npts
C
C ***** CHECK FOR ESCAPE *****
C
        if(nxt .eq. 1) then
          r1 = dsqrt( (q1c - xmu)*(q1c - xmu) + q2c*q2c )
          if(r1 .gt. 5.d0) then
            write (1,*) 'stop 29 - escaped'
            go to 600
          endif
        endif
C
C ***** PERMUTE INDICES *****
C
        nm3 = nm2
        nm2 = nm1
        nm1 = nxt
C
C ***** INTEGRATE ORBIT, HAMING PERMUTES NXT *****
C
        call haming(nxt)
        if(i .eq. 1) go to 500
C
C ***** CALCULATE R DOT V *****
C
        q1d = f(1,nxt)
        q2d = f(3,nxt)
        rdotv(nxt) = (x(1,nxt)-xmu)*q1d + x(3,nxt)*q2d
C
C ***** CHECK FOR PERI/APOAPSE CROSSING *****
C
        if (rdotv(nxt)*rdotv(nm1).gt.0.d0) go to 500
C
C ***** CROSSING HAS OCCURRED!!! *****
C
        INTERPOLATE TO CROSSING TIME
C
        k=k+1
        frac = -rdotv(nxt)/( rdotv(nxt) - rdotv(nm1) )
        q1c = -frac*x(1,nm1) + (1.d0 + frac)*x(1,nxt)
        q2c = -frac*x(3,nm1) + (1.d0 + frac)*x(3,nxt)
        xcross = q1c-xmu
        ycross = q2c

```

```

        write (3,5) xcross,ycross
        write (1,2) i,x(1,nxt),xcross,ycross

500      continue
        write(3,1) 0.0, 0.0
600      continue
        write(2,3) xmu,npts
        write(2,4) xjac,syn

1 format(1x,d16.9,3x,d16.9)
2 format(1x,i6,3(2x,d24.17))
3 format(1x,d24.17,4x,i6)
4 format(1x,d24.17,4x,d24.17)
5 format(2x,2(2x,e23.16))

        close(1)
        close(2)
        close(3)

        stop
        end

        include 'rhs1.for'
        include 'haming.for'
        include 'h.for'

```

```

c **** SAMPLE SECTION INPUT ****
c number of initial conditions
c value of mu
c integration time step / total integration time
c Jacobian constant / sign for conversion to Szebehely's eom
c general output filename
c file for information to be passed to PERIOD
c surface of section plotfile
c Jefferys's x and y initial conditions (enough pairs to equal first number)
c
c
6
0.00095388d0
0.005d0 500.d0
3.15d0 -1.d0
s10out
p10in
s10plot
-0.2d0 0.d0
-0.3d0 0.d0
-0.4d0 0.d0
-0.5d0 0.d0
-0.6d0 0.d0
-0.7d0 0.d0

```

```

C      program period
C
C      ***** "PERIOD" *****
C      - finds symmetric periodic orbits for Restricted problem by
C      finding x and period at zero to force px and y zero at period
C      and while keeping the hamiltonian constant
C
C      implicit double precision (a-h)
C      implicit integer (i-n)
C      implicit double precision (o-z)
C
C      ***** PROGRAM COMMONS *****
C      common /data/ xmu,xmua
C      common /ham/ t,x(20,4),f(20,4),err(20),hh,nn,mode
C
C      dimension xreal(4),ximag(4),cerr(2,1),b(2,2)
C      dimension x(20,4),f(20,4),err(20),xww(2)
C      dimension phi(4,4),xxx(10),rval(2,4),rvec(2,16)
C
C      complex*16 val(4),vec(4,4),ww
C
C      equivalence (val,rval)
C      equivalence (vec,rvec)
C      equivalence (ww,xww)
C
C      character*10 filename1,filename2
C
C      ***** READ INPUT DATA *****
C
C      read(*,*) xmu,npts
C      xmua = 1.d0 - xmu
C      read(*,*) xjac,syn
C      read(*,*) tol,maxit
C      read(*,*) xnot,period
C
C      ***** OPEN OUTPUT FILES *****
C      - file 1 is general output
C      - file 2 is input to program exsyr1
C
C      read(*,*) filename1
C      open(1,FILE=filename1,STATUS='UNKNOWN')
C      read(*,*) filename2
C      open(2,FILE=filename2,STATUS='UNKNOWN')
C
C      ***** REPEAT INPUT VALUES TO DATA FILE *****
C
C      write (1,*) 'mu=',xmu,' 1-mu=',xmua
C      write (1,*) 'max iterations=',maxit,'  tolerance=',tol
C      write (1,*) 'number of points=',npts
C      write (1,*) 'Jacobian constant=',xjac
C
C      write (1,*) 'x0=',xnot,'  period=',period
C      q1 = xnot+xmu
C      write (1,*) 'q1=',q1
C      xham = (xmu*xmua-xjac)/2.d0
C      write (1,*) 'xham=',xham
C
C      ***** BEGIN ITERATION LOOP *****
C
C      do 500 iter = 1,maxit
C
C      ***** SET UP INITIAL STATE *****
C
C      p1 = 0.0d0
C      q2 = 0.0d0
C      r1 = dabs( q1 - xmu )
C      r2 = dabs( q1 + xmua )
C      p2 = q1 + syn*dsqrt( q1*q1 + 2.d0*( xmua/r1
1      + xmu/r2 + xham ))
C
C      ***** INITIAL CONDITIONS *****
C
C      x(1,1) = q1
C      x(2,1) = p1
C      x(3,1) = q2
C      x(4,1) = p2

```

```

C ***** CALCULATE TIMESTEP *****
C
C      hh = period/(dble(npts))
C
C ***** WRITE PROGRESS *****
C
C      write (1,*) 'iteration',iter
C
C ***** INITIALIZE PHI MATRIX *****
C
C      do 40 i = 1,4
C          do 41 j = 1,4
C              ij = 4*i+j
C              x(ij,1) = 0.d0
41          continue
C              x(5*i,1) = 1.d0
40      continue
C
C ***** INITIALIZE INTEGRATION CONSTANTS *****
C
C      mode = 1
C      nn = 20
C      nxt = 0
C      t = 0.d0
C
C ***** INITIALIZE HAMING *****
C
C      call haming(nxt)
C
C      if(nxt .ne. 0) go to 1000
C          write (1,*) 'failure to initialize - stop 99'
C          go to 20
1000      continue
C
C ***** INTEGRATION LOOP *****
C
C      do 1500 i = 1,npts
C          call haming(nxt)
1500      continue
C
C ***** EXTRACT ERROR VECTOR *****
C
C      cerr(1,1) = -x(2,nxt)
C      cerr(2,1) = -x(3,nxt)
C      write (1,*) '      errors'
C      write (1,7) cerr(1,1),cerr(2,1)
C
C ***** CALCULATE CORRECTION MATRIX *****
C
C      dp2dq1 = (p2- xmua*(q1-xmu)/(r1*r1*r1)
C      - xmu*(q1+xmu)/(r2*r2*r2)) / (p2-q1)
1      b(1,1) = x(6,nxt) + x(18,nxt)*dp2dq1
C      b(1,2) = f(2,nxt)
C      b(2,1) = x(7,nxt) + x(19,nxt)*dp2dq1
C      b(2,2) = f(3,nxt)
C
C ***** CALCULATE STATE CORRECTIONS *****
C
C      call leqt2f(b,1,2,2,cerr,idig,xxx,ier)
C
C ***** ADD IN CORRECTIONS *****
C
C      write (1,*) '      corrections'
C      write (1,7) cerr(1,1),cerr(2,1)
C      q1 = q1 + cerr(1,1)
C      period = period + cerr(2,1)
C      write (1,*) '      q1=',q1
C      write (1,*) '      period=',period
C
C ***** CHECK FOR CONVERGENCE *****
C
C      iend = 0
C      if(dabs(cerr(1,1)) .gt. tol) iend = 1
C      if(dabs(cerr(2,1)) .gt. tol) iend = 1
C      if(iend .eq. 0) go to 2000
C
C ***** MAXIMUM ITERATIONS EXCEEDED WITHOUT CONVERGENCE *****

```

```

c
500 continue
   write (1,*) 'Iteration Limit Exceeded - stop 15'
   go to 20
c
c ***** CONVERGED PROCESSING *****
c
2000 continue
   r1 = dabs( q1 - xmu )
   r2 = dabs( q1 + xmu )
   p2 = q1 + syn*dsqrt( q1*q1 + 2.d0*( xmu/r1
1     + xmu/r2 + xham ))
   write(1,*) 'PROGRAM CONVERGED IN',iter,' ITERATIONS'
   write(1,*) '      q1=',q1
   write(1,*) '      p1=',p1
   write(1,*) '      q2=',q2
   write(1,*) '      p2=',p2
   write(1,*) '      period=',period
   pldot = -1.d0*(-1.d0*p2 + xmu*(q1-xmu)/(r1*r1*r1)
1     + xmu*(q1+xmu)/(r2*r2*r2))
   qldot = p1 + q2
   p2dot = -1.d0*(p1 + xmu*q2/(r1*r1*r1)
1     + xmu*q2/(r2*r2*r2))
   q2dot = p2 - q1
   write(1,*) '      qldot=',qldot
   write(1,*) '      pldot=',pldot
   write(1,*) '      q2dot=',q2dot
   write(1,*) '      p2dot=',p2dot
c
c ***** EXTRACT PHI *****
c
   write(1,*) 'phi'
   do 59 i = 5,20
     write(1,*) x(i,nxt)
59 continue
   do 60 i = 1,4
     do 60 j = 1,4
       phi(j,i) = x(4*i+j,nxt)
60 continue
   write(1,*) 'PHI'
   do 61 i = 1,4
     write(1,4) phi(i,1),phi(i,2),phi(i,3),phi(i,4)
61 continue
c
c ***** COMPUTE EIGEN VALUES AND VECTORS OF PHI *****
c
   call devcrg(4,phi,4,val,vec,4)
c
c ***** COMPUTE POINCARÉ EXPONENTS *****
c
   do 80 i = 1,4
     ww = val(i)
c
c     complex log of eigenvalue over period
c
     xreal(i) = dlog(dsqrt(xww(1)*xww(1) + xww(2)*xww(2))) /period
     ximag(i) = datan2( xww(2), xww(1) ) /period
80 continue
   write (1,*) 'EVALUES'
   do 100 i = 1,4
     write(1,5) i,rval(1,i),rval(2,i)
100 continue
   write(1,*) 'POINCARÉ EXPONENTS'
   write(1,*) '      xreal(i)', '      ximag(i)'
   do 120 i = 1,4
     write(1,5) i,xreal(i),ximag(i)
120 continue
   write (1,*) 'EVECTORS'
   do 140 i = 1,16
     if( (i-1)/4 .eq. (i+2)/4 ) then
       write(1,5) (i+3)/4 ,rvec(1,i),rvec(2,i)
     else
       write(1,6) rvec(1,i),rvec(2,i)
     endif
140 continue
c
c ***** OUTPUT DATA FOR NEXT PROGRAM *****
c
   write(2,8) xmu,xjac

```

```

        write(2,3) period,npts
        write(2,2) q1,p1,q2,p2
        write(2,2) q1dot,p1dot,q2dot,p2dot
        do 160 i = 1,4
            write(2,2) phi(i,1),phi(i,2),phi(i,3),phi(i,4)
160    continue
        do 180 i = 1,4
            write(2,6) rval(1,i),rval(2,i)
180    continue
        do 220 i = 1,16
            write(2,6) rvec(1,i),rvec(2,i)
220    continue

        2 format(1x,4(d24.17,2x))
        3 format(1x,d24.17,4x,i6)
        4 format(1x,4(d18.11,2x))
        5 format(1x,i1,2x,2(d24.17,2x))
        6 format(4x,2(d24.17,2x))
        7 format(16x,2(d24.17,2x))
        8 format(1x,2(d24.17,2x))
c
20    continue

        close(1)
        close(2)

        stop
        end

        include 'rhs1.for'
        include 'haming.for'
        include 'h.for'
        include 'leqt2f.for'

```



```

C      program exsyr1
C
C      ***** "EXTENDED, SYMPLECTIC, REAL" *****
C      - program to calculate the extended eigenvector
C      for a repeated root, symplectitize the eigenvector
C      matrix, and create the equivalent real eigenvector and
C      J matrices from the symplectic
C
C      implicit double precision (a-h)
C      implicit integer (i-n)
C      implicit double precision (o-z)
C
C      ***** PROGRAM COMMONS *****
C
C      dimension xww(2),xwc(2),itype(2)
C      dimension phi(4,4),rval(2,4),rvec(2,16)
C
C      complex*16 xxx(10),a(2,2),b(2,1),ww,wc,alpha(2),error
C      complex*16 val(4),fvec(4,4),xj(4,4),evec(4,4),tzvec(4,4)
C      complex*16 evec2(4,4),xj2(4,4)
C
C      equivalence (val,rval)
C      equivalence (fvec,rvec)
C      equivalence (ww,xww)
C      equivalence (wc,xwc)
C
C      character*10 filename1
C      character*10 filename2
C
C      ***** READ INPUT DATA *****
C      - data should be manually rearranged so any degenerate mode
C      eigenvalues and eigenvectors are last
C
C      read(*,*) xmu,xjac
C      read(*,*) period,npts
C      read(*,*) q1,p1,q2,p2
C      read(*,*) qldot,pldot,q2dot,p2dot
C      do 20 i = 1,4
C         read(*,*) phi(i,1),phi(i,2),phi(i,3),phi(i,4)
C 20 continue
C      do 40 i = 1,4
C         read(*,*) rval(1,i),rval(2,i)
C 40 continue
C      do 60 i = 1,16
C         read(*,*) rvec(1,i),rvec(2,i)
C 60 continue
C      read(*,*) itype(1), itype(2)
C      ***** type 0 = degenerate
C      type 1 = positive/negative real
C      type 2 = positive/negative imaginary
C
C      ***** OPEN OUTPUT FILES *****
C      - file 1 is general output
C      - file 2 is input to programs floquet and hamiltonian
C
C      read(*,*) filename1
C      open(1,FILE=filename1,STATUS='UNKNOWN')
C      read(*,*) filename2
C      open(2,FILE=filename2,STATUS='UNKNOWN')
C
C      ***** REPEAT INPUT TO OUTPUT *****
C
C      write(1,*) '      period=',period
C      write(1,*) '      q1=',q1
C
C      ***** FIND POINCARÉ EXPONENTS AND J MATRIX *****
C
C      write(1,*)
C      write(1,*) 'POINCARÉ EXPONENTS'
C      write(1,*) '      REAL          IMAJ'
C      do 80 i = 1,4
C         do 100 j = 1,4
C            xj(j,i) = (0.d0, 0.d0)
C 100 continue
C      ww = val(i)
C

```

```

c      complex log of eigenvalue over period
c
      iii = int((i+1)/2)
      if (itype(iii) .eq. 0) then
        xwc(1) = 0.d0
        xwc(2) = 0.d0
      elseif (itype(iii) .eq. 1) then
        xwc(1) = dlog(dsqrt(xww(1)*xww(1)
&      + xww(2)*xww(2))) /period
        xwc(2) = 0.d0
      elseif (itype(iii) .eq. 2) then
        xwc(1) = 0.d0
        xwc(2) = datan2( xww(2), xww(1) ) /period
      else
        endif
      xj(i,i) = wc
      write(1,2) i,wc
80 continue
      if (itype(2) .eq. 0) xj(3,4)=(1.d0, 0.d0)
c
c ***** assumes no more than one degenerate mode - pair placed second
c
      write(1,*)
      write(1,*) 'J MATRIX'
      write(1,*) '      REAL                      IMAJ'
      do 120 i = 1,4
        do 120 j = 1,4
          if(j .eq. 1) then
            write(1,2) i,xj(j,i)
          else
            write(1,3) xj(j,i)
          endif
        120 continue
c
c ***** FIND EXTENDED EIGENVECTOR IF THERE IS ONE *****
c
      if(itype(2) .ne. 0) goto 140
      a(1,1) = phi(1,1) - val(4)
      a(1,2) = phi(1,4)
      a(2,1) = phi(3,1)
      a(2,2) = phi(3,4)
      b(1,1) = fvec(1,3)*period
      b(2,1) = fvec(3,3)*period
      call cleqt2f(a,1,2,2,b,1,dig,xxx,ier)
      if (ier .eq. 0) go to 160
      write(1,*) 'stop 129 - MATRIX IS SINGULAR'
      goto 2000
160 continue
      fvec(1,4) = b(1,1)
      fvec(2,4) = (0.d0, 0.d0)
      fvec(3,4) = (0.d0, 0.d0)
      fvec(4,4) = b(2,1)
140 write(1,*)
      write(1,*) 'EIGENVECTOR COLUMNS AFTER EXTENDED FOUND (IF NEEDED)'
      write(1,*) '      REAL                      IMAJ'
      do 180 i = 1,4
        do 180 j = 1,4
          if(j .eq. 1) then
            write(1,2) i,fvec(j,i)
          else
            write(1,3) fvec(j,i)
          endif
        180 continue
c
c ***** CHECK TO SEE IF EVECTORS AND EVALUES ARE VALID *****
c
      call check(phi,val,fvec,period,error)
      write(1,*)
      write(1,*) 'max error in evalue and evector check'
      write(1,*) 'error=',error
c
c ***** FIND SYMPLECTIC EIGENVECTORS *****
c
      call symplec(fvec,tzvec,error)
      write(1,*)
      write(1,*) 'max error in symplectic F^T*Z*F'
      write(1,*) 'error=',error
      ww = error

```

```

        if (dabs(xww(1)) .gt. 1.d-10) goto 200
        if (dabs(xww(2)) .gt. 1.d-10) goto 200
        goto 220
c
c ***** GET MULTIPLICATION FACTORS AND CORRECT F MATRIX *****
c
200 alpha(1) = (1.d0, 0.d0)/cdsqrt(tzvec(1,2))
   alpha(2) = (1.d0, 0.d0)/cdsqrt(tzvec(3,4))
   write(1,*) 'alpha1=',alpha(1)
   write(1,*) 'alpha2=',alpha(2)
   do 240 i = 1,4
      evec(i,1) = fvec(i,1)*alpha(1)
      evec(i,2) = fvec(i,2)*alpha(1)
      evec(i,3) = fvec(i,3)*alpha(2)
      evec(i,4) = fvec(i,4)*alpha(2)
240 continue
   call symplec(evec,tzvec,error)
   write(1,*)
   write(1,*) 'max error in symplectic E^T*Z*E'
   write(1,*) 'error=',error
   goto 260
220 do 280 i = 1,4
      do 280 j = 1,4
         evec(j,i) = fvec(j,i)
280 continue
260 write(1,*)
   write(1,*) 'SYMPLECTIC EIGENVECTOR COLUMNS'
   write(1,*) '      REAL                      IMAJ'
   do 300 i = 1,4
      do 300 j = 1,4
         if(j .eq. 1) then
            write(1,2) i,evec(j,i)
         else
            write(1,3) evec(j,i)
         endif
300 continue
c
c ***** FIND REAL E AND J MATRICES *****
c
c      call real(evec,xj,evec2,xj2,itype)
c
c ***** CHECK IF STILL SYMPLECTIC *****
c
c      call symplec(evec2,tzvec,error)
c      write(1,*)
c      write(1,*) 'max error in symplectic after E made real'
c      write(1,*) 'error=',error
c      write(1,*)
c      write(1,*) 'SYMPLECTIC REAL EIGENVECTOR COLUMNS'
c      write(1,*) '      REAL                      IMAJ'
c      do 320 i = 1,4
c         do 320 j = 1,4
c            if(j .eq. 1) then
c               write(1,2) i,evec2(j,i)
c            else
c               write(1,3) evec2(j,i)
c            endif
320 continue
c      write(1,*)
c      write(1,*) 'REAL J MATRIX COLUMNS'
c      write(1,*) '      REAL                      IMAJ'
c      do 340 i = 1,4
c         do 340 j = 1,4
c            if(j .eq. 1) then
c               write(1,2) i,xj2(j,i)
c            else
c               write(1,3) xj2(j,i)
c            endif
340 continue
c
c ***** OUTPUT FOR NEXT PROGRAM *****
c
c      write(2,3) xmu,xjac
c      write(2,6) period,npts
c      write(2,7) q1,p1,q2,p2
c      write(2,8) itype(1),itype(2)
c      do 400 i = 1,4
c         do 400 j = 1,4

```

```

        write(2,7) evec2(j,i)
400 continue
    do 420 i = 1,4
        do 420 j = 1,4
            write(2,7) xj2(j,i)
420 continue
2000 continue

2 format(1x,i1,2(2x,d24.17))
3 format(2x,2(2x,d24.17))
5 format(1x,d24.17)
6 format(1x,d24.17,4x,i6)
7 format(1x,4(2x,d24.17))
8 format(1x,i2,2x,i2)

close(1)
close(2)

stop
end

include 'cleqt2f.for'
include 'symplectic.for'
include 'check.for'
include 'real.for'

```

```

c ***** SAMPLE EXSYRL INPUT *****
c mu / Jacobian constant
c period / number of timesteps
c initial values of q1, p1, q2, p2
c initial values of qldot, pldot, q2dot, p2dot
c phi by row
c real & imaginary components of eigenvalues
c real & imaginary components of eigenvectors (matched in order with eigenvalues)
c types of Poicare exponent pairs
c general output file
c file for data to be passed to FLOQUET and HAMILTONIAN
c
c
0.953880000000000005D-03 0.315D+01
0.62565804113518846D+01 4000
-0.5127081520827D+00 0.0000000000000D+00 0.0000000000000D+00 -0.1516521888144D+01
0.0000000000000D+00 0.2265881088747D+01 -0.1003813736061D+01 0.0000000000000D+00
0.8609790735321D+00 -0.1755076922110D-01 -0.3961686779363D-01 -0.5556742445356D-01
0.1686687698163D+03 0.8609790734677D+00 -0.3138081071241D+00 0.6775180018322D+02
-0.6775180018329D+02 0.5556742447948D-01 0.1125430816271D+01 -0.2722871816792D+02
0.3138081069484D+00 0.3961686779099D-01 0.8942606411584D-01 0.1125430816198D+01
0.98640988974257526D+00 -0.16430316317961827D+00
0.98640988974257526D+00 0.16430316317961827D+00
1.0D+00 0.0000000000000000D+00
1.0D+00 0.0000000000000000D+00
-0.70221866516062548D-12 -0.10442224677270372D-01
0.10000000000000000D+01 0.0000000000000000D+00
-0.39970546784512651D+00 -0.76641470947436119D-12
0.17951039960051318D-11 0.23570946055484081D-01
-0.70221866516062548D-12 0.10442224677270372D-01
0.10000000000000000D+01 0.0000000000000000D+00
-0.39970546784512651D+00 0.76641470947436119D-12
0.17951039960051318D-11 -0.23570946055484081D-01
0.0000000000000000D+00 0.d0
0.22658810887474097D+01 0.d0
-0.10038137360616357D+01 0.d0
0.0000000000000000D+00 0.d0
-0.17366902140924529D-05 0.0000000000000000D+00
0.10000000000000000D+01 0.0000000000000000D+00
-0.44301253975986643D+00 0.0000000000000000D+00
0.43449248273103301D-05 0.0000000000000000D+00
2 0
ex10out
f10in

```

```

C      program floquet
C
C      ***** "FLOQUET" *****
C      - turns state elements and elements of E into sets
C      of 100 Fourier coefficients in order to create
C      exact solution
C
C      implicit double precision (a-h)
C      implicit integer (i-n)
C      implicit double precision (o-z)
C
C      ***** PROGRAM COMMONS *****
C
C      common /data/ xmu,xmua
C      common /ham/ t,x(20,4),f(20,4),err(20),hh,nn,mode
C      common /fdat/ xj(4,4)
C
C      dimension x(20,4),f(20,4),err(20),xww(2)
C      dimension s(4,200),v(4,4,200),itype(2)
C      dimension ck(101),sk(101),fxn(200)
C      dimension vec(4,4),vecf(4,4),diff(4,4),xj(4,4)
C
C      character*10 filename1,filename2
C
C      ***** READ INPUT DATA *****
C
C      read(*,*) xmu,xjac
C      xmua = 1.d0 - xmu
C      read(*,*) period, npts
C      hh = period/(dble(npts))
C      itrip = npts/200
C      read(*,*) q1o,p1o,q2o,p2o
C      read(*,*) itype(1),itype(2)
C
C      ***** itest makes certain all elements of E and J are real
C
C      itest = 0
C      do 40 i = 1,4
C        do 40 j = 1,4
C          read(*,*) xww(1),xww(2)
C          if (xww(2) .gt. 1.d-10) itest = 1
C          vec(j,i) = xww(1)
C      40 continue
C      do 60 i = 1,4
C        do 60 j = 1,4
C          read(*,*) xww(1),xww(2)
C          if (xww(2) .gt. 1.d-10) itest = 1
C          xj(j,i) = xww(1)
C      60 continue
C      if (itest .eq. 0) goto 65
C      write(1,*) 'There is an imaginary component of E or J'
C      write(1,*) 'stop'
C      goto 2000
C      65 continue
C
C      ***** OPEN OUTPUT FILES *****
C      - file 1 is general output
C      - file 2 is input to program exact
C
C      read(*,*) filename1
C      open(1,FILE=filename1,STATUS='UNKNOWN')
C      read(*,*) filename2
C      open(2,FILE=filename2,STATUS='UNKNOWN')
C
C      ***** REPEAT INPUT VALUES TO DATA FILE *****
C
C      write(1,*) '      xmu=',xmu
C      write(1,*) '      xjac=',xjac
C      write(1,*) '      period=',period
C      write(1,*) '      points=',npts
C      write(1,*) '      trip=',itrip
C      write(1,*) '      timestep=',hh
C      write(1,*) '      initial conditions'
C      write(1,*) '      q1o=',q1o
C      write(1,*) '      p1o=',p1o
C      write(1,*) '      q2o=',q2o
C      write(1,*) '      p2o=',p2o

```

```

      write(1,*) '          E MATRIX BY COLUMN'
      do 80 i = 1,4
        do 80 j = 1,4
          if(j .eq. 1) then
            write(1,2) i,vec(j,i)
          else
            write(1,3) vec(j,i)
          endif
60      continue
      write(1,*) '          J MATRIX BY COLUMN'
      do 100 i = 1,4
        do 100 j = 1,4
          if(j .eq. 1) then
            write(1,2) i,xj(j,i)
          else
            write(1,3) xj(j,i)
          endif
80      continue
100     continue
c
c ***** SET UP INITIAL STATE *****
c
      x(1,1) = q1o
      x(2,1) = p1o
      x(3,1) = q2o
      x(4,1) = p2o
c
c ***** INITIALIZE E AT t(o) *****
c
      do 120 i = 1,4
        do 120 j = 1,4
          x(i*4+j,1) = vec(j,i)
120     continue
      mode = 1
      nn = 20
      nxt = 0
      t = 0.d0
c
c ***** INITIALIZE HAMING *****
c
      call haming(nxt)
      if(nxt .ne. 0) goto 140
      write(1,*) 'FAILURE TO INITIALIZE - STOP 99'
      write(1,7) f(1,1),f(2,1)
      write(1,7) f(3,1),f(4,1)
      goto 2000
140     continue
c
c ***** BEGIN INTEGRATION LOOP *****
c
      do 160 i = 1,200
        do 180 j = 1,4
          s(j,i) = x(j,nxt)
          do 180 k = 1,4
            v(k,j,i) = x((j*4)+k,nxt)
180         continue
        do 200 ii = 1,itrip
          call haming(nxt)
200         continue
160     continue
c
c ***** FEED STATE/EVECTORS TO FOURIER *****
c
      write(2,5) xmu
      write(2,6) period,npts
      write(2,8) itype(1),itype(2)
      xnot = q1o - xmu
      ynot = q2o
      write(2,7) xnot,ynot
      write(2,*) 'insert increase/decrease to xnot,xjac'
      write(2,7) xjac, -1.d0
      write(2,*) 'insert itrip value here'
      write(2,*) 'insert two filenames here'
      do 220 i = 1,4
        do 240 j = 1,200
          fxn(j) = s(i,j)
240     continue
      call fourier(fxn,ck,sk,100)
      do 260 j = 1,100

```

```

        write(2,7) ck(j),sk(j)
260    continue
220  continue
    do 280 i = 1,4
        do 280 j = 1,4
            do 300 k = 1,200
                fxn(k) = v(j,i,k)
300            continue
            call fourier(fxn,ck,sk,100)
            do 320 k = 1,100
                write(2,7) ck(k),sk(k)
320            continue
280  continue
c
c ***** EXTRACT FINAL STATE *****
c
    q1t = x(1,nxt)
    p1t = x(2,nxt)
    q2t = x(3,nxt)
    p2t = x(4,nxt)
    do 340 i = 1,4
        do 340 j = 1,4
            vecf(j,i) = x(i*4+j,nxt)
340  continue
c
c ***** FINAL STATE CONDITIONS *****
c
    write(1,*) 'STATE AT Tf'
    write(1,*) '      q1t=',q1t
    write(1,*) '      p1t=',p1t
    write(1,*) '      q2t=',q2t
    write(1,*) '      p2t=',p2t
    write(1,*) '      E MATRIX BY COLUMN'
    do 360 i = 1,4
        do 360 j = 1,4
            if(j .eq. 1) then
                write(1,2) i,vecf(j,i)
            else
                write(1,3) vecf(j,i)
            endif
            diff(j,i) = vecf(j,i)-vec(j,i)
360  continue
c
c ***** DIFFERENCE IN INITIAL AND FINAL CONDITIONS *****
c
    write(1,*)
    write(1,*)
    write(1,*) '      REAL E(t)-E(0)'
    do 400 i = 1,4
        do 400 j = 1,4
            if(j .eq. 1) then
                write(1,2) i,diff(j,i)
            else
                write(1,3) diff(j,i)
            endif
400  continue

2  format(1x,i1,2x,d24.17)
3  format(4x,d24.17)
5  format(2x,d24.17)
6  format(2x,d24.17,2x,i6)
7  format(2x,2(2x,d24.17))
8  format(2x,i2,4x,i2)

2000 continue

    close(1)
    close(2)

    stop
    end

    include 'rhs2.for'
    include 'haming.for'
    include 'h.for'
    include 'fourier.for'

```



```

c      program hamiltonian
c
c      ***** "HAMILTONIAN" *****
c      - turns periodic coefficients needed to expand hamiltonian
c      into sets of 100 fourier coefficients
c
c      implicit double precision (a-h)
c      implicit integer (i-n)
c      implicit double precision (o-z)
c
c      ***** PROGRAM COMMONS *****
c      common /data/ xmu,xmua
c      common /ham/ t,x(20,4),f(20,4),err(20),hh,nn,mode
c      common /fdat/ xj(4,4)
c
c      dimension x(20,4),f(20,4),err(20),xww(2)
c      dimension ck(101),sk(101),fxn(200)
c      dimension c(20,200),s(4),itype(2),w(2)
c      dimension vec(4,4),vecf(4,4),diff(4,4),xj(4,4)
c      dimension e(4,4),xh3(4,4,4),tc(4,4,4)
c
c      equivalence (ww,xww)
c      character*10 filename1,filename2
c
c      ***** READ INPUT DATA *****
c
c      read(*,*) xmu
c      xmua = 1.d0 - xmu
c      read(*,*) period, npts
c      itrip = npts/200
c      hh = period/(dble(npts))
c      read(*,*) q1o,p1o,q2o,p2o
c      read(*,*) itype(1),itype(2)
c
c      ***** itest makes certain all elements of E and J are real
c
c      itest = 0
c      do 40 i = 1,4
c        do 40 j = 1,4
c          read(*,*) xww(1),xww(2)
c          if (xww(2) .gt. 1.d-10) itest = 1
c          vec(j,i) = xww(1)
c        40 continue
c      do 60 i = 1,4
c        do 60 j = 1,4
c          read(*,*) xww(1),xww(2)
c          if (xww(2) .gt. 1.d-10) itest = 1
c          xj(j,i) = xww(1)
c        60 continue
c      if (itest .eq. 0) goto 65
c      write(1,*) 'There is an imaginary component of E or J'
c      write(1,*) 'stop'
c      goto 2000
c      65 continue
c
c      ***** OPEN OUTPUT FILES *****
c      - file 1 is general output
c      - file 2 is input to program expand
c
c      read(*,*) filename1
c      open(1,FILE=filename1,STATUS='UNKNOWN')
c      read(*,*) filename2
c      open(2,FILE=filename2,STATUS='UNKNOWN')
c
c      ***** REPEAT INPUT VALUES TO DATA FILE *****
c
c      write(1,*) '      xmu=',xmu
c      write(1,*) '      xjac=',xjac
c      write(1,*) '      period=',period
c      write(1,*) '      points=',npts
c      write(1,*) '      trip=',itrip
c      write(1,*) '      timestep=',hh
c      write(1,*) 'initial conditions'
c      write(1,*) '      q1o=',q1o
c      write(1,*) '      p1o=',p1o
c      write(1,*) '      q2o=',q2o

```

```

write(1,*) '      p2o=',p2o
write(1,*) '      E MATRIX BY COLUMN'
do 80 i = 1,4
  do 80 j = 1,4
    if(j .eq. 1) then
      write(1,2) i,vec(j,i)
    else
      write(1,3) vec(j,i)
    endif
80 continue
write(1,*) '      J MATRIX BY COLUMN'
do 100 i = 1,4
  do 100 j = 1,4
    if(j .eq. 1) then
      write(1,2) i,xj(j,i)
    else
      write(1,3) xj(j,i)
    endif
100 continue
C
C ***** SET UP INITIAL STATE *****
C
  x(1,1) = q1o
  x(2,1) = p1o
  x(3,1) = q2o
  x(4,1) = p2o
C
C ***** INITIALIZE F AT t(o) *****
C
  do 120 i = 1,4
    do 120 j = 1,4
      x(i*4+j,1) = vec(j,i)
120 continue
  mode = 1
  nn = 20
  nxt = 0
  t = 0.d0
C
C ***** INITIALIZE HAMING *****
C
  call haming(nxt)
  if(nxt .ne. 0) goto 140
  write(1,*) 'FAILURE TO INITIALIZE - STOP 99'
  write(1,7) f(1,1),f(2,1)
  write(1,7) f(3,1),f(4,1)
  goto 2000
140 continue
C
C ***** BEGIN INTEGRATION LOOP *****
C
  do 200 i = 1,200
    do 220 j = 1,4
      s(j) = x(j,nxt)
      do 220 k = 1,4
        e(k,j) = x(j*4+k,nxt)
220      continue
C
C ***** COMPUTE THIRD ORDER TENSOR *****
C
  do 240 j = 1,4
    do 240 k = 1,4
      do 240 m = 1,4
        xh3(j,k,m) = h(s,3,j,k,m,0,0)
        tc(j,k,m) = 0.d0
240      continue
C
C ***** COMPUTE PERIODIC COEFFICIENTS *****
C
C      - first loop variable tc -
C
  do 260 j = 1,4
    do 260 k = 1,4
      do 260 m = 1,4
C
C      - second loop variable xh3 -
C
  do 260 jj = 1,4
    do 260 kk = 1,4
      do 260 mm = 1,4

```

```

tc(j,k,m) = tc(j,k,m) +
      xh3(jj,kk,mm)*e(jj,j)*e(kk,k)*e(mm,m)
&
260 continue
c(1,i) = tc(1,1,1)/6.d0
c(2,i) = (tc(1,1,2)+tc(1,2,1)+tc(2,1,1))/6.d0
c(3,i) = (tc(1,1,3)+tc(1,3,1)+tc(3,1,1))/6.d0
c(4,i) = (tc(1,1,4)+tc(1,4,1)+tc(4,1,1))/6.d0
c(5,i) = (tc(1,2,2)+tc(2,1,2)+tc(2,2,1))/6.d0
c(6,i) = (tc(1,2,3)+tc(1,3,2)+tc(2,1,3)+tc(2,3,1)
      +tc(3,1,2)+tc(3,2,1))/6.d0
&
c(7,i) = (tc(1,2,4)+tc(1,4,2)+tc(2,1,4)+tc(2,4,1)
      +tc(4,1,2)+tc(4,2,1))/6.d0
&
c(8,i) = (tc(1,3,3)+tc(3,1,3)+tc(3,3,1))/6.d0
c(9,i) = (tc(1,3,4)+tc(1,4,3)+tc(3,1,4)+tc(3,4,1)
      +tc(4,1,3)+tc(4,3,1))/6.d0
&
c(10,i) = (tc(1,4,4)+tc(4,1,4)+tc(4,4,1))/6.d0
c(11,i) = tc(2,2,2)/6.d0
c(12,i) = (tc(2,2,3)+tc(2,3,2)+tc(3,2,2))/6.d0
c(13,i) = (tc(2,2,4)+tc(2,4,2)+tc(4,2,2))/6.d0
c(14,i) = (tc(2,3,3)+tc(3,2,3)+tc(3,3,2))/6.d0
c(15,i) = (tc(2,3,4)+tc(2,4,3)+tc(3,2,4)+tc(3,4,2)
      +tc(4,2,3)+tc(4,3,2))/6.d0
&
c(16,i) = (tc(2,4,4)+tc(4,2,4)+tc(4,4,2))/6.d0
c(17,i) = tc(3,3,3)/6.d0
c(18,i) = (tc(3,3,4)+tc(3,4,3)+tc(4,3,3))/6.d0
c(19,i) = (tc(3,4,4)+tc(4,3,4)+tc(4,4,3))/6.d0
c(20,i) = tc(4,4,4)/6.d0
do 360 j = 1,itrip
      call haming(nxt)
360 continue
200 continue
c
c ***** COMPUTE FOURIER COEFFICIENTS FROM PERIODIC ONES *****
c
      write(2,6) period,npts
      write(2,*) 'insert modal initial displacements here'
      write(2,*) 'insert itrip value here'
      do 380 i = 1,3,2
        iii = int((i+1)/2)
        if (itype(iii) .eq. 2) then
          w(iii) = xj(i,i+1)
        else
          w(iii) = xj(i,i)
        endif
      endif
380 continue
      write(2,7) w(1),w(2)
      write(2,8) itype(1),itype(2)
      write(2,*) 'insert two filenames here'
      do 400 i = 1,20
        do 420 j = 1,200
          fxn(j) = c(i,j)
420 continue
          call fourier(fxn,ck,sk,100)
          do 440 j = 1,100
            write(2,7) ck(j),sk(j)
440 continue
400 continue
c
c ***** EXTRACT FINAL STATE *****
c
      qlt = x(1,nxt)
      plt = x(2,nxt)
      q2t = x(3,nxt)
      p2t = x(4,nxt)
      do 480 i = 1,4
        do 480 j = 1,4
          vecf(j,i) = x(i*4+j,nxt)
480 continue
c
c ***** FINAL STATE CONDITIONS *****
c
      write(1,*) 'STATE AT Tf'
      write(1,*) '      qlt=',qlt
      write(1,*) '      plt=',plt
      write(1,*) '      q2t=',q2t
      write(1,*) '      p2t=',p2t
      write(1,*) '      E MATRIX BY COLUMN'
      do 500 i = 1,4

```

```

        do 500 j = 1,4
            if(j .eq. 1) then
                write(1,2) i,vecf(j,i)
            else
                write(1,3) vecf(j,i)
            endif
            diff(j,i) = vec(j,i)-vecf(j,i)
500 continue
        write(1,*) '          REAL E(t)-E(0)'
        do 520 i = 1,4
            do 520 j = 1,4
                if(j .eq. 1) then
                    write(1,2) i,diff(j,i)
                else
                    write(1,3) diff(j,i)
                endif
520 continue

        2 format(1x,i1,2x,d24.17)
        3 format(4x,d24.17)
        6 format(2x,d24.17,2x,i6)
        7 format(2x,2(2x,d24.17))
        8 format(2x,i2,4x,i2)

2000 continue

        close(1)
        close(2)

        stop
        end

        include 'rhs2.for'
        include 'haming.for'
        include 'h.for'
        include 'fourier.for'

```



```

c      program exact
c
c      ***** "EXACT" *****
c      - integrates a nearly periodic orbit, subtracts
c      the periodic reference, transforms the
c      result into modal variables, and creates a plotfile
c      of the modal variables b1, b2, b3, and b4
c
c      implicit double precision (a-h)
c      implicit integer (i-n)
c      implicit double precision (o-z)
c
c      ***** PROGRAM COMMONS *****
c
c      common /data/ xmu,xmua
c      common /ham/ t,x(20,4),f(20,4),err(20),hh,nn,mode
c
c      dimension x(20,4),f(20,4),err(20)
c      dimension cf(20),sinn(100),coss(100)
c      dimension ck(20,100),sk(20,100),dx(4,1)
c      dimension e(4,4),xxx(50),itype(2)
c
c      equivalence (ww,xww)
c
c      character*10 filename1,filename2
c
c      ***** read input data *****
c
c      read(*,*) xmu
c      xmua = 1.d0 - xmu
c      read(*,*) period,npts
c      hh = period/(dble(npts))
c      read(*,*) itype(1),itype(2)
c      read(*,*) xnot,ynot
c      read(*,*) xd,xjacd
c      xnot = xnot+xd
c      read(*,*) xjac,syn
c      xjac = xjac + xjacd
c      read(*,*) itrip
c
c      ***** OPEN OUTPUT FILES *****
c      - file 1 is general output
c      - file 2 is a plotfile of the exact solution
c
c      read(*,*) filename1
c      open(1,FILE=filename1,STATUS='UNKNOWN')
c      read(*,*) filename2
c      open(2,FILE=filename2,STATUS='UNKNOWN')
c
c      do 60 i = 1,20
c         do 60 j = 1,100
c            read(*,*) ck(i,j),sk(i,j)
c         60 continue
c
c      ***** GET q1,p1,q2,p2 FOR GIVEN x0,y0, AND JACOBIAN *****
c
c      q1 = xnot+xmu
c      q2 = ynot
c      xham = (xmu*xmua-xjac)/2.d0
c      r1 = dsqrt((q1-xmu)*(q1-xmu) + q2*q2)
c      r2 = dsqrt((q1+xmua)*(q1+xmua) + q2*q2)
c      d = xham + xmua/r1 + xmu/r2
c      g = q2/(q1-xmu)
c      a = 0.5d0*(g*g + 1.d0)
c      b = -(g*g*xmu + g*q2 + q1)
c      c = 0.5d0*g*g*xmu*xmu + g*q2*xmu - d
c      disc = b*b - 4.d0*a*c
c      if( disc .lt. 0.d0) then
c         write (1,*) 'stop 10 - no roots'
c         go to 2000
c      endif
c      p2 = (-b+syn*dsqrt(disc))/(2.d0*a)
c      p1 = g*(xmu-p2)
c
c      ***** REPEAT INPUT VALUES TO DATA FILE *****
c
c      write(1,*) 'mu=',xmu,' 1-mu=',xmua

```

```

        write(1,*) 'step=',hh,' period=',period
        write(1,*) 'number of points=',npts
        write(1,*)
        write(1,*) 'initial conditions (Jefferys)'
        write(1,*) 'x0=',xnot,' y0=',ynot
        write(1,*) 'Jacobian constant=',xjac
        write(1,*)
        write(1,*) 'initial conditions (Szebehely)'
        write(1,*) 'q1=',q1,' p1=',p1
        write(1,*) 'q2=',q2,' p2=',p2
        write(1,*) 'Hamiltonian constant=',xham
c
        mode = 0
        nn = 4
        nxt = 0
        t = 0.d0
c
        pi = dacos(-1.d0)
        w0 = (2.d0*pi)/period
        write(1,*) ' w0=',w0
c
c ***** INITIAL CONDITIONS *****
c
        x(1,1) = q1
        x(2,1) = p1
        x(3,1) = q2
        x(4,1) = p2
c
c ***** INITIALIZE HAMILTONIAN *****
c
        call haming(nxt)
c
c ***** TURN OFF SECOND EOM EVALUATION *****
c
        nxt = -nxt
        if(nxt .ne. 0) go to 499
        write(1,*) 'stop 99 - unable to start Haming'
        go to 2000
499 continue
c
c ***** INTEGRATION LOOP *****
c
        do 500 i = 0,250
c
c ***** compute sin(n*theta), cos(n*theta), n=1,50 *****
c
                coss(1) = acos(w0*t)
                sinn(1) = dsin(w0*t)
                coss(2) = 2.d0*coss(1)*coss(1)-1.d0
                sinn(2) = 2.d0*sinn(1)*coss(1)
                do 520 j = 3,100
                        coss(j) = 2.d0*coss(j-1)*coss(1)-coss(j-2)
                        sinn(j) = 2.d0*sinn(j-1)*coss(1)-sinn(j-2)
520                continue
c
c ***** REASSEMBLE PERIODIC TRAJ AND EIGENVECTOR MATRIX *****
c
                do 540 j = 1,20
                        cf(j) = ck(j,1)
                        do 540 k = 1,99
                                cf(j) = cf(j) + ck(j,k+1)*coss(k)
                                + sk(j,k+1)*sinn(k)
540                continue
                do 560 j = 1,4
                        dx(j,1) = x(j,iabs(nxt))-cf(j)
560                continue
c
c ***** PLACE EIGENVECTORS IN 4X4 MATRIX FOR INVERSION *****
c
                do 580 j = 1,4
                        do 580 k = 1,4
                                e(k,j) = cf(j*4+k)
580                continue
c
c *** CALCULATE DELTA b WITH DELTA b = EJECTORS-1 * DELTA x ***
c
                idig = 0
                call leqt2f(e,1,4,4,dx,idig,xxx,ier)

```

```

        if (i .ne. 0) goto 620
        write(1,*)
        write(1,*) 'Initial Modal Conditions'
        write(1,*) '      b1 - b4'
        write(1,5) dx(1,1)
        write(1,5) dx(2,1)
        write(1,5) dx(3,1)
        write(1,5) dx(4,1)
620      continue
        write(2,6) t,dx(1,1),dx(2,1),dx(3,1),dx(4,1)
        do 640 j = 1,itrip
            call haming(nxt)
640      continue
500 continue
c ***** EXTRACT FINAL CONDITIONS *****
c
        write(1,*)
        write(1,*)
        write(1,*) 'final state minus initial state (Szebehely)'
        write(1,*) '      q1,p1,q2,p2'
        write(1,*) x(1,nxt)-q1
        write(1,*) x(2,nxt)-p1
        write(1,*) x(3,nxt)-q2
        write(1,*) x(4,nxt)-p2
2000 continue

        2 format(2x,2(2x,d24.17))
        4 format(4x,d12.5,2x,d24.17)
        5 format(4x,d24.17)
        6 format(2x,5(2x,e23.16))

        close(1)
        close(2)
        close(3)
        close(4)

        stop
        end

        include 'rhs1.for'
        include 'haming.for'
        include 'h.for'
        include 'leqt2f.for'

```

```

c ***** SAMPLE EXACT INPUT *****
c mu
c period / number of integration steps(for calculation of timestep)
c types of Poincare exponent pairs
c initial value of x and y (Jefferys eom initial conditions)
c amount of change to x / amount of change to Jacobian constant
c initial Jacobian constant / conversion sign
c integration steps between sampled data(250 total data samples)
c general output
c plot file of time and modal variables
c 20 sets of 100 sine and cosine Fourier representation pairs
c
c
0.9538800000000000D-03
0.62565804113518473D+01      4000
2      -1
-0.51366203208274741D+00      0.0000000000000000D+00
0.d0  0.d0
3.15d0  -1.d0
630
fqlout
exaplot1
0.14272865159998943D-02      0.0000000000000000D+00
-0.44658422312811241D+00      0.50812964946800321D-13
0.95570615732216700D-04      0.15916670759175133D-12
-0.58321183772268116D-01      -0.24715715585266195D-13
etc.... (a total of 2000 pairs of numbers)

```

```

C
C      program expanded
C
C      ***** "EXPANDED" *****
C      - Using the periodic coefficients made by the program
C      hamiltonian, the eom for the truncated
C      hamiltonian case are integrated. A plotfile of modal
C      variables b1, b2,b3, and b4 is created.
C
C      implicit double precision (a-h)
C      implicit integer (i-n)
C      implicit double precision (o-z)
C
C      ***** PROGRAM COMMONS *****
C
C      common /datarhs/ w0,w1,w2,ck(20,100),sk(100,50),itype(2)
C      common /ham/ t,x(20,4),f(20,4),err(20),hh,nn,mode
C
C      dimension x(20,4),f(20,4),err(20),itype(2)
C      dimension ck(20,100),sk(20,100),b0(4),b(4)
C
C      character*10 filename1,filename2
C
C      ***** READ INPUT DATA *****
C
C      read(*,*) period,npts
C      hh = period/(dble(npts))
C      do 40 i = 1,4
C          read(*,*) b0(i)
C      40 continue
C      read(*,*) itrip
C      read(*,*) w1,w2
C      read(*,*) itype(1),itype(2)
C
C      ***** OPEN OUTPUT FILES *****
C      - file 1 is general output
C      - file 2 is a plotfile for the expanded case
C
C      read(*,*) filename1
C      open(1,FILE=filename1,STATUS='UNKNOWN')
C      read(*,*) filename2
C      open(2,FILE=filename2,STATUS='UNKNOWN')
C
C      do 60 i = 1,20
C          do 60 j = 1,100
C              read(*,*) ck(i,j),sk(i,j)
C      60 continue
C
C      ***** OUTPUT INPUTS *****
C
C      write(1,*) ' orbit period=',period
C      write(1,*) ' # of points=',npts
C      write(1,*) ' timestep=',hh
C      write(1,*) ' initial state (MODAL)'
C      write(1,*) '      b1=',b0(1)
C      write(1,*) '      b2=',b0(2)
C      write(1,*) '      b3=',b0(3)
C      write(1,*) '      b4=',b0(4)
C
C      ***** SET UP INITIAL STATE *****
C
C      do 80 i = 1,4
C          x(i,1) = b0(i)
C      80 continue
C
C      mode = 0
C      nn = 4
C      nxt = 0
C      t = 0.d0
C
C      pi = dacos(-1.d0)
C      w0 = 2.d0*pi/period
C
C      ***** INITIALIZING HAMING *****
C
C      call haming(nxt)
C      if(nxt .ne. 0) go to 199

```

```

        write (1,*) 'stop 99 - unable to start Haming'
        go to 2000
199 continue
c
c ***** BEGIN INTEGRATION LOOP *****
c
        do 220 i = 1,250
            do 240 j = 1,4
                b(j) = x(j,nxt)
240         continue
                write(2,4) t,b(1),b(2),b(3),b(4)
                do 300 j = 1,itrip
                    call haming(nxt)
300         continue
220 continue
c
c ***** EXTRACT FINAL STATE *****
c
        do 340 j = 1,4
            b(j) = x(j,nxt)
340 continue
            write(1,*)
            write(1,*) 'final state (Modal)'
            write(1,*) '      b1=',b(1)
            write(1,*) '      b2=',b(2)
            write(1,*) '      b3=',b(3)
            write(1,*) '      b4=',b(4)

2000 continue

2 format (2x,2(2x,d24.17))
3 format (4x,d12.5,2x,d24.17)
4 format (2x,5(2x,e23.16))

        close(1)
        close(2)

        stop
        end

        include 'haming.for'
        include 'rhs3.for'

```

```

c ***** SAMPLE EXPANDED INPUT *****
c period / number of integration steps (used to determine the timestep)
c four initial modal displacements (found in the general output of EXACT)
c integration steps between sampled data (250 total data samples)
c value of omega for Poincare exponent pairs (0.0 in degenerate case)
c types of Poincare exponent pairs
c general output
c plot file of time and modal variables
c 20 sets of 100 sine and cosine Fourier representation pairs
c
c
0.62565804113518473D+01    4000
0.39599531533278684D-12
-0.19873186154673566D-15
-0.14793686800523243D-13
-0.10107539548798371D-13
630
-0.2638047004D-01    0.00000000000D+00
2    -1
-0.26380470049156036D-01
2
hmlout
expplot1
0.54375306107848423D-01    0.000000000000000D+00
0.11386208422001964D-02    0.18637175136504425D-13
0.73473610830684946D-01    0.23774386834207739D-10
-0.54706262117310002D-04    -0.13772226414854315D-12
-0.93120150715344754D-01    0.17098303259355240D-10
etc.... (a total of 2000 pairs of numbers)

```

```

c      subroutine haming(nxt)
c
c      haming is an ordinary differential equations integrator
c      it is a fourth order predictor-corrector algorithm
c      which means that it carries along the last four
c      values of the state vector, and extrapolates these
c      values to obtain the next value (the prediction part)
c      and then corrects the extrapolated value to find a
c      new value for the state vector.
c
c      the value nxt in the call specifies which of the 4 values
c      of the state vector is the "next" one.
c      nxt is updated by haming automatically, and is zero on
c      the first call
c
c      the user supplies an external routine rhs(nxt) which
c      evaluates the equations of motion
c
c      common /ham/ x,y(20,4),f(20,4),errest(20),h,n,mode
c      double precision x,y,f,errest,h,hh,xo,tol
c      implicit double precision (a-h)
c      implicit integer (i-n)
c      implicit double precision (o-z)
c
c      dimension y(20,4),f(20,4),errest(20)
c
c      all of the good stuff is in this common block.
c      x is the independent variable ( time )
c      y(6,4) is the state vector- 4 copies of it, with nxt
c      pointing at the next one
c      f(6,4) are the equations of motion, again four copies
c      a call to rhs(nxt) updates an entry in f
c      errest is an estimate of the truncation error - normally not
c      used
c      n is the number of equations being integrated - 6 or 42 here
c      h is the time step
c      mode is 0 for just EOM, 1 for both EOM and EOv
c
c      tol = 0.0000000001d+00
c      switch on starting algorithm or normal propagation
c      if(nxt) 190,10,200
c
c      this is hamings starting algorithm....a predictor - corrector
c      needs 4 values of the state vector, and you only have one- the
c      initial conditions.
c      haming uses a Picard iteration (slow and painful) to get the
c      other three.
c      if it fails, nxt will still be zero upon exit, otherwise
c      nxt will be 1, and you are all set to go
c
10  xo = x
   hh = h/2.0d+00
   call rhs(1)
   do 40 i = 2,4
     x = x + hh
     do 20 i = 1,n
20  y(i,1) = y(i,1-1) + hh*f(i,1-1)
     call rhs(1)
     x = x + hh
     do 30 i = 1,n
30  y(i,1) = y(i,1-1) + h*f(i,1)
40  call rhs(1)
     jsw = -10
50  isw = 1
     do 120 i = 1,n
       hh = y(i,1) + h*( 9.0d+00*f(i,1) + 19.0d+00*f(i,2)
1   - 5.0d+00*f(i,3) + f(i,4) ) / 24.0d+00
       if( dabs( hh - y(i,2) ) .lt. tol ) go to 70
       isw = 0
70  y(i,2) = hh
       hh = y(i,1) + h*( f(i,1) + 4.0d+00*f(i,2) + f(i,3) )/3.0d+00
       if( dabs( hh-y(i,3) ) .lt. tol ) go to 90
       isw = 0
90  y(i,3) = hh
       hh = y(i,1) + h*( 3.0d+00*f(i,1) + 9.0d+00*f(i,2) + 9.0d+00*f(i,3)
1   + 3.0d+00*f(i,4) ) / 8.0d+00
       if( dabs( hh-y(i,4) ) .lt. tol ) go to 110

```

```

      isw = 0
110 y(i,4) = hh
120 continue
      x = xo
      do 130 l = 2,4
      x = x + h
130 call rhs(l)
      if(isw) 140,140,150
140 jsw = jsw + 1
      if(jsw) 50,280,280
150 x = xo
      isw = 1
      jsw = 1
      do 160 i = 1,n
160 errest(i) = 0.0d0
      nxt = 1
      go to 280
190 jsw = 2
      nxt = iabs(nxt)
c
c      this is hamings normal propagation loop -
c
200 x = x + h
      npl = mod(nxt,4) + 1
      go to (210,230),isw
c      permute the index nxt modulo 4
210 go to (270,270,270,220),nxt
220 isw = 2
230 nm2 = mod(npl,4) + 1
      nml = mod(nm2,4) + 1
      npo = mod(nml,4) + 1
c
c      this is the predictor part
c
      do 240 i = 1,n
      f(i,nm2) = y(i,npl) + 4.0d+00*h*( 2.0d+00*f(i,npo) - f(i,nml)
1      + 2.0d+00*f(i,nm2) ) / 3.0d+00
240 y(i,npl) = f(i,nm2) - 0.925619835d0*errest(i)
c
c      now the corrector - fix up the extrapolated state
c      based on the better value of the equations of motion
c
      call rhs(npl)
      do 250 i = 1,n
      y(i,npl) = ( 9.0d+00*y(i,npo) - y(i,nm2) + 3.0d+00*h*( f(i,npl)
1      + 2.0d+00*f(i,npo) - f(i,nml) ) ) / 8.0d+00
      errest(i) = f(i,nm2) - y(i,npl)
250 y(i,npl) = y(i,npl) + 0.0743801653d0 * errest(i)
      go to (260,270),jsw
260 call rhs(npl)
270 nxt = npl
280 return
      end

```

```

c      function h(x,iord,i,j,k,l,m)
c
c      restricted problem in canonical coordinates
c
c      state vector x = ( q1, p1, q2, p2 )
c
c      dimension x(4)
c      common /data/ xmu,xmua
c      implicit double precision (a-h)
c      implicit integer (i-n)
c      implicit double precision (o-z)
c
c      preliminaries
c
c      qa = x(1) - xmu
c      qb = x(1) + xmua
c      r1 = (qa*qa + x(3)*x(3))**.5d0
c      r2 = (qb*qb + x(3)*x(3))**.5d0
c
c      branch on order
c
c      jord = iord + 1
c      go to (1, 1000, 2000, 3000),jord
c
c      *****
c      **                                     **
c      **          Order  Zero              **
c      **                                     **
c      *****
c
c      1 continue
c      h = 0.5d0*(x(2)*x(2) + x(4)*x(4)) + x(3)*x(2) - x(1)*x(4)
c      1      - xmu/r1 - xmu/r2
c      return
c
c      1000 continue
c
c      *****
c      **                                     **
c      **          Order  One               **
c      **                                     **
c      *****
c
c      r13 = r1**3.d0
c      r23 = r2**3.d0
c      go to (1001, 1002, 1003, 1004), i
c
c      1001 h = -x(4) + xmua*qa/r13 + xmu*qb/r23
c      return
c      1002 h = x(2) + x(3)
c      return
c      1003 h = x(2) + xmua*x(3)/r13 + xmu*x(3)/r23
c      return
c      1004 h = x(4) - x(1)
c      return
c
c      2000 continue
c
c      *****
c      **                                     **
c      **          Order  Two              **
c      **                                     **
c      *****
c
c      r13 = r1**3.d0
c      r23 = r2**3.d0
c      r15 = r1**5.d0
c      r25 = r2**5.d0
c
c      go to (2001, 2002, 2003, 2004),i
c      2001 go to (2011, 2012, 2013, 2014),j
c      2002 go to (2021, 2022, 2023, 2024),j
c      2003 go to (2031, 2032, 2033, 2034),j
c      2004 go to (2041, 2042, 2043, 2044),j
c
c      2011 h = xmua/r13 + xmu/r23 -3.d0*xmua*qa*qa/r15
c      1      -3.d0*xmu*qb*qb/r25
c      return

```

```

2012 h = 0.d0
      return
2013 h = -3.d0*xmua*qa*x(3)/r15 - 3.d0*xmu*qb*x(3)/r25
      return
2014 h = -1.d0
      return
c
2021 h = 0.d0
      return
2022 h = 1.d0
      return
2023 h = 1.d0
      return
2024 h = 0.d0
      return
c
2031 go to 2013
2032 h = 1.d0
      return
2033 h = -3.d0*xmua*x(3)*x(3)/r15 - 3.d0*xmu*x(3)*x(3)/r25
      1 + xmua/r13 + xmu/r23
      return
2034 h = 0.d0
      return
c
2041 h = -1.d0
      return
2042 h = 0.d0
      return
2043 h = 0.d0
      return
2044 h = 1.d0
      return
c
3000 continue
c
c *****
c **                                     **
c **           Order   Three           **
c **                                     **
c *****
c
      r15 = r1**5.d0
      r25 = r2**5.d0
      r17 = r1**7.d0
      r27 = r2**7.d0
c
      go to (30001, 30002, 30003, 30004),i
30001 go to (30110, 30120, 30130, 30140),j
30002 go to (30210, 30220, 30230, 30240),j
30003 go to (30310, 30320, 30330, 30340),j
30004 go to (30410, 30420, 30430, 30440),j
c note matrix is quite sparse now.....
30110 go to (30111, 30112, 30113, 30114),k
30130 go to (30131, 30132, 30133, 30134),k
30310 go to (30311, 30312, 30313, 30314),k
30330 go to (30331, 30332, 30333, 30334),k
c
30111 h = -9.d0*xmua*qa/r15 - 9.d0*xmu*qb/r25
      1 + 15.d0*xmua*qa*qa/r17 + 15.d0*xmu*qb*qb/r27
      return
30112 h = 0.d0
      return
30113 h = -3.d0*xmua*x(3)/r15 - 3.d0*xmu*x(3)/r25
      1 + 15.d0*xmua*qa*qa*x(3)/r17 + 15.d0*xmu*qb*qb*x(3)/r27
      return
30114 h = 0.d0
      return
c
30120 h = 0.d0
      return
c
30131 go to 30113
30132 h = 0.d0
      return
30133 h = -3.d0*xmua*qa/r15 - 3.d0*xmu*qb/r25
      1 + 15.d0*xmua*qa*x(3)*x(3)/r17 + 15.d0*xmu*qb*x(3)*x(3)/r27
      return

```

```

30134 h = 0.d0
      return
c
30140 h = 0.d0
      return
30210 h = 0.d0
      return
30220 h = 0.d0
      return
30230 h = 0.d0
      return
30240 h = 0.d0
      return
c
30311 go to 30113
30312 h = 0.d0
      return
30313 go to 30133
30314 h = 0.d0
      return
c
30320 h = 0.d0
      return
c
30331 go to 30133
30332 h = 0.d0
      return
30333 h = -9.d0*xmua*x(3)/r15 - 9.d0*xmu*x(3)/r25
      1 + 15.d0*(xmua/r17 + xmu/r27)*x(3)*x(3)*x(3)
      return
c
30334 h = 0.d0
      return
30340 h = 0.d0
      return
30410 h = 0.d0
      return
30420 h = 0.d0
      return
30430 h = 0.d0
      return
30440 h = 0.d0
      return
c
      end

```

```

c      subroutine leqt2f(a,m,n,nn,b,idgt,x,ier)
c
c      gaussian elimination with maximal pivoting
c      interface simulates IMSL routine
c      solution of a system of linear equations for m right sides
c      a: matrix of system
c      m: number of rhs
c      n: order of a, rows in b
c      ia: row dimension of a,b
c      b: right hand sides....solution on return
c      idgt: ignored here....in imsl 0=no acc test on input
c           idgt= #digits ok on output in imsl
c      x: in imsl, n**2 + 3*n
c      ier: 129: singular matrix, 0=ok
c
c      dimension a(nn,nn),b(nn,m),irr(50),x(1)
c      double precision a,b,x,anorm,amax,p,tol
c
c      find max norm of a
c      anorm = 0.d0
c      do 5 i = 1,n
c         do 5 j = 1,n
c            if(dabs(a(i,j)) .gt. anorm) anorm = dabs(a(i,j))
c      5 continue
c      set tolerance = 2** (- number of binary digits in mantissa)
c      tol = 1.d-12
c      ier = 0
c      id = 1
c      do 10 i = 1,n
c         irr(i) = 0
c      10 continue
c      20 ir = 1
c         is = 1
c         amax = 0.d0
c      c      find max pivot
c      do 60 i = 1,n
c         if(irr(i)) 60,30,60
c      30      do 50 j = 1,n
c         p = dabs(a(i,j))
c         if(p-amax) 50,50,40
c      40      ir = i
c         is = j
c         amax = p
c      50      continue
c      60 continue
c      singularity test
c      if(amax/anorm .gt. tol) go to 70
c      ier = 129
c      go to 120
c      forward elimination
c      70 irr(ir) = is
c         do 90 i = 1,n
c            if(i .eq. ir) go to 90
c            p = a(i,is)/a(ir,is)
c            do 80 j = 1,n
c               a(i,j) = a(i,j) - p*a(ir,j)
c      80      continue
c            a(i,is) = 0.0
c            do 85 j = 1,m
c               b(i,j) = b(i,j) - p*b(ir,j)
c      85      continue
c      90 continue
c      id = id + 1
c      if(id .le. n) go to 20
c      back substitution
c      c
c      do 115 j = 1,m
c         do 100 i = 1,n
c            ir = irr(i)
c            x(ir) = b(i,j)/a(i,ir)
c      100      continue
c            do 110 i = 1,n
c               b(i,j) = x(i)
c      110      continue
c      115 continue
c      120 return
c      end

```

```

c      subroutine cleqt2f(a,m,n,nn,b,idgt,x,ier)
c
c      complex version of
c      gaussian elimination with maximal pivoting
c      interface simulates IMSL routine
c      solution of a system of linear equations for m right sides
c      a: matrix of system
c      m: number of rhs
c      n: order of a, rows in b
c      ia: row dimension of a,b
c      b: right hand sides....solution on return
c      idgt: ignored here....in imsl 0=no acc test on input
c      idgt= #digits ok on output in imsl
c      x: in imsl, n**2 + 3*n
c      ier: 129: singular matrix, 0=ok
c
c      implicit double precision (a-h)
c      implicit integer (i-n)
c      implicit double precision (o-z)
c      complex*16 a(nn,nn),b(nn,m),x(28),check,d
c      dimension irr(50),rcheck(2)
c      equivalence (check,rcheck)
c
c      find max norm of a
c      anorm = 0.d0
c      do 5 i = 1,n
c         do 5 j = 1,n
c            check = a(i,j)
c            temp = dsqrt(rcheck(1)*rcheck(1)
c              1      +rcheck(2)*rcheck(2))
c            if(temp .gt. anorm) anorm = temp
c      5 continue
c      set tolerance = 2** (- number of binary digits in mantissa)
c      tol = 1.d-12
c      ier = 0
c      id = 1
c      do 10 i = 1,n
c         irr(i) = 0
c      10 continue
c      20 ir = 1
c      is = 1
c      amax = 0.d0
c      find max pivot
c      do 60 i = 1,n
c         if(irr(i)) 60,30,60
c      30      do 50 j = 1,n
c         check = a(i,j)
c         p = dsqrt(rcheck(1)*rcheck(1)
c           1      +rcheck(2)*rcheck(2))
c         if(p-amax) 50,50,40
c      40      ir = i
c         is = j
c         amax = p
c      50      continue
c      60 continue
c      singularity test
c      if(amax/anorm .gt. tol) go to 70
c      ier = 129
c      go to 120
c      forward elimination
c      70 irr(ir) = is
c      do 90 i = 1,n
c         if(i .eq. ir) go to 90
c         d = a(i,is)/a(ir,is)
c         do 80 j = 1,n
c            a(i,j) = a(i,j) - d*a(ir,j)
c      80      continue
c         a(i,is) = 0.0
c         do 85 j = 1,m
c            b(i,j) = b(i,j) - d*b(ir,j)
c      85      continue
c      90 continue
c      id = id + 1
c      if(id .le. n) go to 20
c      back substitution
c      do 115 j = 1,m
c         do 100 i = 1,n

```

```

        ir = irr(i)
        x(ir) = b(i,j)/a(i,ir)
100      continue
        do 110 i = 1,n
            b(i,j) = x(i)
110      continue
115      continue
120      return
        end

```

```

C      subroutine check(phi,val,vec,period,error)
C
C      - checks the transformation from phi to jordan normal form
C
      implicit double precision (a-h)
      implicit integer (i-n)
      implicit double precision (o-z)
      dimension xww(2),xwc(2),phi(4,4)
      complex*16 ww,wc,ejt(4,4),vec(4,4),val(4)
      complex*16 fe(4,4),pf(4,4),pi(4,4),piv(4,4)
      equivalence (ww,xww)
      equivalence (wc,xwc)
C
C      ***** CALCULATE e^(Jt) *****
C
      do 100 i = 1,4
        do 120 j = 1,4
          ejt(j,i) = (0.d0, 0.d0)
120      continue
          ejt(i,i) = val(i)
100    continue
          if(ejt(3,3) .eq. (1.d0, 0.d0)) then
            xww(1) = period
            xww(2) = 0.d0
            ejt(3,4) = ww
          else
            endif
C
C      ***** FIND phi*f - f*e^(Jt) *****
C
      do 200 i = 1,4
        do 200 j = 1,4
          fe(i,j) = (0.d0, 0.d0)
          pf(i,j) = (0.d0, 0.d0)
          do 200 k = 1,4
            fe(i,j) = fe(i,j) + vec(i,k)*ejt(k,j)
            pf(i,j) = pf(i,j) + phi(i,k)*vec(k,j)
200      continue
          error = (0.d0, 0.d0)
          ww = error
          do 220 i = 1,4
            do 220 j = 1,4
              wc = pf(j,i) - fe(j,i)
              if(dabs(xwc(1)) .gt. dabs(xww(1))) xww(1) = xwc(1)
              if(dabs(xwc(2)) .gt. dabs(xww(2))) xww(2) = xwc(2)
220      continue
C
C      ***** FIND (phi-e^(Jt)*I)*vec *****
C
      do 300 i = 1,4
        do 320 j = 1,4
          do 340 k = 1,4
            pi(k,j) = phi(k,j)
            piv(j,k) = (0.d0, 0.d0)
340          continue
            pi(j,j) = phi(j,j) - val(i)
320          continue
            do 360 j = 1,4
              do 360 k = 1,4
                piv(j,i) = piv(j,i) + pi(j,k)*vec(k,i)
360          continue
300      continue
          if(ejt(3,4) .eq. (0.d0, 0.d0)) goto 400
          piv(2,4) = piv(2,4) - vec(2,3)*period
          piv(3,4) = piv(3,4) - vec(3,3)*period
          do 400 i = 1,4
            do 400 j = 1,4
              wc = piv(j,i)
              if(dabs(xwc(1)) .gt. dabs(xww(1))) xww(1) = xwc(1)
              if(dabs(xwc(2)) .gt. dabs(xww(2))) xww(2) = xwc(2)
400      continue
          error = ww

      2 format(1x,i1,2(2x,d24.17))
      3 format(2x,2(2x,d24.17))
      return
      end

```

```

c      subroutine symplec(vec,tzvec,error)
c
c      - checks if eigenvector matrix is symplectic
c
c      implicit double precision (a-h)
c      implicit integer (i-n)
c      implicit double precision (o-z)
c
c      dimension z(4,4),xww(2),xwc(2)
c
c      complex*16 vec(4,4),tvec(4,4),zvec(4,4),tzvec(4,4)
c      complex*16 error,ww,wc
c
c      equivalence(ww,xww)
c      equivalence(wc,xwc)
c
c      data z/ 0.d0, -1.d0, 0.d0, 0.d0,
1          1.d0, 0.d0, 0.d0, 0.d0,
2          0.d0, 0.d0, 0.d0, -1.d0,
3          0.d0, 0.d0, 1.d0, 0.d0/
c
c      ***** TRANSPOSE vec, STORE AS tvec *****
c
c      do 440 i = 1,4
c          do 440 j = 1,4
c              tvec(i,j) = vec(j,i)
c      440 continue
c
c      ***** CALCULATE Z VEC *****
c
c      do 480 i = 1,4
c          do 480 j = 1,4
c              zvec(i,j) = (0.d0, 0.d0)
c              do 480 k = 1,4
c                  zvec(i,j) = zvec(i,j) + z(i,k)*vec(k,j)
c      480 continue
c
c      T
c      ***** CALCULATE VEC * Z VEC = Z *****
c
c      do 500 i = 1,4
c          do 500 j = 1,4
c              tzvec(i,j) = (0.d0, 0.d0)
c              do 500 k = 1,4
c                  tzvec(i,j) = tzvec(i,j) + tvec(i,k)*zvec(k,j)
c      500 continue
c
c      calculate max error
c
c      error = (0.d0, 0.d0)
c      ww = error
c      do 600 i = 1,4
c          do 600 j = 1,4
c              wc = tzvec(j,i) - z(j,i)
c              if(dabs(xwc(1)) .gt. dabs(xww(1))) xww(1) = xwc(1)
c              if(dabs(xwc(2)) .gt. dabs(xww(2))) xww(2) = xwc(2)
c      600 continue
c      error = ww
c
c      2 format(1x,i1,2(2x,d24.17))
c      3 format(2x,2(2x,d24.17))
c
c      return
c      end

```

```

C      subroutine real(vec,xj,vecnew,xjnew,itype)
C
C      - converts imaginary E and J to real E and J
C
C      implicit double precision (a-h)
C      implicit integer (i-n)
C      implicit double precision (o-z)
C
C      dimension xwc(2),xww(2),itype(2)
C
C      complex*16 vec(4,4),xj(4,4),vecnew(4,4),xjnew(4,4)
C      complex*16 ww,wc,t(4,4),ti(4,4),temp(4,4)
C      complex*16 vec2(4,4),xj2(4,4)
C
C      equivalence (wc,xwc)
C      equivalence (ww,xww)
C
C      ***** REMOVE IMAGINARY POINCARÉ EXPONENTS *****
C
C      ***** CALCULATE T AND T INVERSE *****
C
C      do 20 i = 1,4
C        do 25 j = 1,4
C          t(i,j) = (0.d0, 0.d0)
C          ti(i,j) = (0.d0, 0.d0)
C25      continue
C        t(i,i) = (1.d0, 0.d0)
C        ti(i,i) = (1.d0, 0.d0)
C20      continue
C      do 40 i = 1,3,2
C        i1i = int((i+1)/2)
C        if (itype(i1i).eq. 2) then
C          t(i,i) = (.5d0, -.5d0)
C          t(i,i+1) = (.5d0, -.5d0)
C          t(i+1,i) = (-.5d0, -.5d0)
C          t(i+1,i+1) = (.5d0, .5d0)
C          ti(i,i) = (.5d0, .5d0)
C          ti(i,i+1) = (-.5d0, .5d0)
C          ti(i+1,i) = (.5d0, .5d0)
C          ti(i+1,i+1) = (.5d0, -.5d0)
C        else
C          endif
C40      continue
C
C      ***** CALCULATE NEW E AND J MATRICES *****
C
C      do 60 i = 1,4
C        do 60 j = 1,4
C          temp(i,j) = (0.d0, 0.d0)
C          do 60 k = 1,4
C            temp(i,j) = temp(i,j) + t(i,k)*xj(k,j)
C60      continue
C      do 80 i = 1,4
C        do 80 j = 1,4
C          vec2(i,j) = (0.d0, 0.d0)
C          xj2(i,j) = (0.d0, 0.d0)
C          do 80 k = 1,4
C            vec2(i,j) = vec2(i,j) + vec(i,k)*ti(k,j)
C            xj2(i,j) = xj2(i,j) + temp(i,k)*ti(k,j)
C80      continue
C
C      ***** REMOVE IMAGINARY EIGENVECTORS *****
C
C      ***** CALCULATE T AND T INVERSE *****
C
C      do 100 i = 1,4
C        do 105 j = 1,4
C          t(i,j) = (0.d0, 0.d0)
C          ti(i,j) = (0.d0, 0.d0)
C105      continue
C        t(i,i) = (1.d0, 0.d0)
C        ti(i,i) = (1.d0, 0.d0)
C100      continue
C      do 120 i = 1,3,2
C        itest = 0
C        do 140 j = 1,4
C          wc = vec2(j,i)

```

```

        if (dabs(xwc(2)) .gt. 1.d-10) itest = 1
140    continue
        if (itest .eq. 1) then
            t(i,i) = (0.d0, 0.d0)
            t(i,i+1) = (0.d0, 1.d0)
            t(i+1,i) = (0.d0, 1.d0)
            t(i+1,i+1) = (0.d0, 0.d0)
            ti(i,i) = (0.d0, 0.d0)
            ti(i,i+1) = (0.d0, -1.d0)
            ti(i+1,i) = (0.d0, -1.d0)
            ti(i+1,i+1) = (0.d0, 0.d0)
            iii = int((i+1)/2)
            if(itype(iii) .eq. 0) itype(iii) = -1
            else
            endif
120 continue
c
c ***** CALCULATE NEW EIGENVECTOR MATRIX AND NEW J MATRIX *****
c
        do 180 i = 1,4
            do 180 j = 1,4
                temp(i,j) = (0.d0, 0.d0)
                do 180 k = 1,4
                    temp(i,j) = temp(i,j) + t(i,k)*xj2(k,j)
180 continue
            do 200 i = 1,4
                do 200 j = 1,4
                    vecnew(i,j) = (0.d0, 0.d0)
                    xjnew(i,j) = (0.d0, 0.d0)
                    do 200 k = 1,4
                        vecnew(i,j) = vecnew(i,j) + vec2(i,k)*ti(k,j)
                        xjnew(i,j) = xjnew(i,j) + temp(i,k)*ti(k,j)
200 continue
        return
end

```

```

C      subroutine fourier(f,ck,sk,n)
C
C      - subroutine to create set of 100 fourier coefficients for a
C      given periodic variable
C
C      implicit double precision (a-h)
C      implicit integer (i-n)
C      implicit double precision (o-z)
C
C      dimension f(200),ck(101),sk(101)
C
C      pi = dacos(-1.d0)
C      twopi = 2.d0*pi
C      alpha = twopi/dble(2*n)
C      n2m1 = 2*n-1
C
C      ***** ORDER K LOOP *****
C
C      do 500 k = 0,n
C
C      ***** COSINE SUM *****
C
C      ck(k+1) = 0.d0
C      do 200 j = 0,n2m1
C      ck(k+1) = ck(k+1) + f(j+1)*dcos(dble(k*j)*alpha)
200      continue
C      ck(k+1) = ck(k+1)/db1e(n)
C
C      ***** SINE SUM *****
C
C      if(k .eq. 0) goto 500
C      if(k .eq. n) goto 500
C      sk(k+1) = 0.d0
C      do 400 j = 1,n2m1
C      sk(k+1) = sk(k+1) + f(j+1)*dsin(dble(k*j)*alpha)
400      continue
C      sk(k+1) = sk(k+1)/db1e(n)
500      continue
C
C      ***** CORRECT FIRST AND LAST COSINE COEFFICIENT *****
C
C      ck(1) = .5d0*ck(1)
C      ck(n+1) = .5d0*ck(n+1)
C
C      return
C      end

```

```

C      subroutine rhs(k)
C
C      canonical EOM and EOV, 4th order system
C      Phi matrix stored by cols, end to end
C
C      common /data/ xmu, xmua
C      common /ham/ t,x(20,4),f(20,4),err(20),hh,nn,mode
C
C      external h
C
C      implicit double precision (a-h)
C      implicit integer (i-n)
C      implicit double precision (o-z)
C
C      dimension x(20,4),f(20,4),err(20),xx(4),z(4,4),grdh(4,4),A(4,4)
C
C      data z/ 0.d0, -1.d0, 0.d0, 0.d0,
1          1.d0, 0.d0, 0.d0, 0.d0,
2          0.d0, 0.d0, 0.d0, -1.d0,
3          0.d0, 0.d0, 1.d0, 0.d0/
C
C      ***** EXTRACT STATE *****
C
C      do 10 i = 1,4
10  xx(i) = x(i,k)
C
C      ***** EQUATIONS OF MOTION *****
C
C      f(1,k) = h(xx,1,2,0,0,0,0)
C      f(2,k) = -h(xx,1,1,0,0,0,0)
C      f(3,k) = h(xx,1,4,0,0,0,0)
C      f(4,k) = -h(xx,1,3,0,0,0,0)
C
C      if(mode .eq. 0) return
C
C      ***** CALCULATE ORDER 2 GRADIENT MATRIX *****
C
C      do 20 i = 1,4
C        do 20 j = 1,4
C          grdh(i,j) = h(xx,2,i,j,0,0,0)
20  continue
C
C      ***** MATRIX MPY BY Z *****
C
C      do 30 i = 1,4
C        do 30 ii = 1,4
C          A(i,ii) = 0.d0
C          do 30 j = 1,4
C            A(i,ii) = A(i,ii) + z(i,j)*grdh(j,ii)
30  continue
C
C      ***** CALCULATE A PHI *****
C
C      row loop
C      do 35 i = 1,4
C        col loop
C        do 35 ii = 1,4
C          ij = 4*ii+i
C          f(ij,k) = 0.d0
C          do 35 j = 1,4
C            f(ij,k) = f(ij,k) + A(i,j)*x(4*ii+j,k)
35  continue
C      return
C      end

```

```

c      subroutine rhs(k)
c
c      canonical EOM and EOV, 4th order system
c      Phi matrix stored by cols, end to end
c
c      common /data/ xmu,xmua
c      common /ham/ t,x(20,4),f(20,4),err(20),hh,nn,mode
c      common /fdat/ xj(4,4)
c
c      external h
c
c      implicit double precision (a-h)
c      implicit integer (i-n)
c      implicit double precision (o-z)
c
c      dimension x(20,4),f(20,4),err(20),xx(4),z(4,4)
c      dimension grdh(4,4),a(4,4),vec(4,4)
c      dimension xj(4,4),fdot(4,4)
c      dimension f1(4,4),f2(4,4)
c      data z/ 0.d0, -1.d0, 0.d0, 0.d0,
1         1.d0, 0.d0, 0.d0, 0.d0,
2         0.d0, 0.d0, 0.d0, -1.d0,
3         0.d0, 0.d0, 1.d0, 0.d0/
c ***** EXTRACT STATE *****
c
c      do 10 i = 1,4
c         xx(i) = x(i,k)
10 continue
c ***** EQUATIONS OF MOTION *****
c
c      f(1,k) = h(xx,1,2,0,0,0,0)
c      f(2,k) = -h(xx,1,1,0,0,0,0)
c      f(3,k) = h(xx,1,4,0,0,0,0)
c      f(4,k) = -h(xx,1,3,0,0,0,0)
c
c      if(mode .eq. 0) return
c ***** CALCULATE ORDER 2 GRADIENT MATRIX *****
c
c      do 20 i = 1,4
c         do 20 j = 1,4
c            grdh(i,j) = h(xx,2,i,j,0,0,0)
20 continue
c ***** MATRIX MPY BY Z *****
c
c      do 30 i = 1,4
c         do 30 ii = 1,4
c            a(i,ii) = 0.d0
c            do 30 j = 1,4
c               a(i,ii) = a(i,ii) + z(i,j)*grdh(j,ii)
30 continue
c ***** CALCULATE A*F AND F*J *****
c
c      do 33 i = 1,4
c         do 33 j = 1,4
c            vec(j,i) = x(i*4+j,k)
33 continue
c      row loop
c      do 35 i = 1,4
c         col loop
c         do 35 j = 1,4
c            f1(i,j) = 0.d0
c            f2(i,j) = 0.d0
c            do 35 ii = 1,4
c               f1(i,j) = f1(i,j) + a(i,ii)*vec(ii,j)
c               f2(i,j) = f2(i,j) + vec(i,ii)*xj(ii,j)
35 continue
c      do 36 i = 1,4
c         do 36 j = 1,4
c            fdot(i,j) = f1(i,j)-f2(i,j)
36 continue
c      do 37 i = 1,4
c         do 37 j = 1,4
c            f(i*4+j,k) = fdot(j,i)
37 continue
c
c      return
c      end

```

```

c      subroutine rhs(k)
c
c      calculate rhs for nearly-periodic eom, using
c      expanded hamiltonian
c
c      canonical EOM and EOv, 4th order system
c
c      common /datarhs/ w0,w1,w2,ck(20,100),sk(20,100),itype(2)
c      common /ham/ t,x(20,4),f(20,4),err(20),hh,nn,mode
c
c      implicit double precision (a-h)
c      implicit integer (i-n)
c      implicit double precision (o-z)
c
c      dimension x(20,4),f(20,4),err(20)
c      dimension coss(100),sinn(100),itype(2),w(2)
c      dimension ck(20,100),sk(20,100),c(20),b(4),bdot(4)
c      w(1) = w1
c      w(2) = w2
c
c      ***** GENERATE SIN(1 to 50 *w0)AND COS(1 to 50 * w0) *****
c
c      coss(1) = dcos(w0*t)
c      sinn(1) = dsin(w0*t)
c      coss(2) = 2.d0*coSS(1)*coSS(1)-1.d0
c      sinn(2) = 2.d0*sinn(1)*coSS(1)
c      do 100 i = 3,100
c          coss(i) = 2.d0*coSS(i-1)*coSS(1)-coSS(i-2)
c          sinn(i) = 2.d0*sinn(i-1)*coSS(1)-sinn(i-2)
c      100 continue
c
c      ***** RECONSTRUCT PERIODIC FUNCTION COEFFICIENTS *****
c
c      do 200 i = 1,20
c          c(i) = ck(i,1)
c          do 200 j = 1,99
c              c(i) = c(i) + ck(i,j+1)*coSS(j) + sk(i,j+1)*sinn(j)
c      200 continue
c
c      ***** ESTABLISH MODAL VECTORS *****
c
c      do 240 i = 1,4
c          b(i) = x(i,k)
c      240 continue
c
c      ***** CALCULATE BDOT AND ESTABLISH EOM *****
c
c      bdot(1) = +1.d0*b(1)*b(1)*c(2) + 2.d0*b(1)*b(2)*c(5)
c      & + 1.d0*b(1)*b(3)*c(6) + 1.d0*b(1)*b(4)*c(7)
c      & + 3.d0*b(2)*b(2)*c(11) + 2.d0*b(2)*b(3)*c(12)
c      & + 2.d0*b(2)*b(4)*c(13) + 1.d0*b(3)*b(3)*c(14)
c      & + 1.d0*b(3)*b(4)*c(15) + 1.d0*b(4)*b(4)*c(16)
c      bdot(2) = -3.d0*b(1)*b(1)*c(1) - 2.d0*b(1)*b(2)*c(2)
c      & - 2.d0*b(1)*b(3)*c(3) - 2.d0*b(1)*b(4)*c(4)
c      & - 1.d0*b(2)*b(2)*c(5) - 1.d0*b(2)*b(3)*c(6)
c      & - 1.d0*b(2)*b(4)*c(7) - 1.d0*b(3)*b(3)*c(8)
c      & - 1.d0*b(3)*b(4)*c(9) - 1.d0*b(4)*b(4)*c(10)
c      bdot(3) = 1.d0*b(1)*b(1)*c(4) + 1.d0*b(1)*b(2)*c(7)
c      & + 1.d0*b(1)*b(3)*c(9) + 2.d0*b(1)*b(4)*c(10)
c      & + 1.d0*b(2)*b(2)*c(13) + 1.d0*b(2)*b(3)*c(15)
c      & + 2.d0*b(2)*b(4)*c(16) + 1.d0*b(3)*b(3)*c(18)
c      & + 2.d0*b(3)*b(4)*c(19) + 3.d0*b(4)*b(4)*c(20)
c      bdot(4) = -1.d0*b(1)*b(1)*c(3) - 1.d0*b(1)*b(2)*c(6)
c      & - 2.d0*b(1)*b(3)*c(8) - 1.d0*b(1)*b(4)*c(9)
c      & - 1.d0*b(2)*b(2)*c(12) - 2.d0*b(2)*b(3)*c(14)
c      & - 1.d0*b(2)*b(4)*c(15) - 3.d0*b(3)*b(3)*c(17)
c      & - 2.d0*b(3)*b(4)*c(18) - 1.d0*b(4)*b(4)*c(19)
c      do 250 i = 1,3,2
c          i11 = int((i+1)/2)
c          if (itype(i11) .eq. -1) then
c              bdot(i+1) = bdot(i+1) + b(i)
c          elseif (itype(i11) .eq. 0) then
c              bdot(i) = bdot(i) + b(i+1)
c          elseif (itype(i11) .eq. 1) then
c              bdot(i) = bdot(i) + w(i11)*b(i)
c              bdot(i+1) = bdot(i+1) - w(i11)*b(i+1)
c          elseif (itype(i11) .eq. 2) then

```

```

        bdot(i) = bdot(i) + w(iii)*b(i+1)
        bdot(i+1) = bdot(i+1) - w(iii)*b(i)
    else
    endif
250 continue
do 260 i = 1,4
    f(i,k) = bdot(i)
260 continue

return
end

```

Bibliography

- Brouwer, Dirk and Gerald M. Clemence. *Methods of Celestial Mechanics*. New York and London: Academic Press, 1961.
- Calico, Robert A. and William E. Wiesel. "Control of Time-Periodic System," *Journal of Guidance, Control, and Dynamics*, 7: 671-676 (November-December 1984).
- Jefferys, William H. *An Atlas of Surface of Section for the Restricted Problem of Three Bodies*. University of Texas at Austin: Applied Mechanics Research Laboratory, 1971.
- Pars, L.A. *A Treatise on Analytical Dynamics*. New York: John Wiley and Sons, 1965.
- Reid, Gary J. *Linear System Fundamentals* (Continuous and Discrete, Classic and Modern). New York: McGraw-Hill Publishing Company, 1983.
- Ross, Capt David A. *Perturbation Theory for Restricted Three-Body Orbits*. MS thesis, AFIT/GA/ENY/91D-7. School of Engineering, Air Force Institute of Technology(AU), Wright-Patterson AFB OH, December 1991.
- Siegel, Carl L. and Jürgen K. Moser. *Lectures on Celestial Mechanics*. Translation by C.I. Kalme. Berlin, Heidelberg, and New York: Spriger-Verlag, 1971.
- Strang, Gilbert. *Linear Algebra and Its Applications*. San Diego: Harcourt Brace Jovanovich, Publishers, 1988.
- Szebehely, Victor. *Theory of Orbits* (The Restricted Problem of Three Bodies). New York and London: Academic Press, 1967.
- Wiesel, William E. "Perturbation Theory in the Vicinity of a Periodic Orbit by Repeated Linear Transformations," *Journal of Celestial Mechanics*, 23: 231-242 (1981).
- Wiesel, William E. *Spaceflight Dynamics*. New York: McGraw-Hill Book Company, 1989.
- Wiesel, William E. and David J. Pohlen. "Canonical Floquet Theory." Article for Journal Submission (October 1992).

Vita

Capt David J. Pohlen was born on 22 September 1965 in Minneapolis Minnesota. After graduating from Benilde-St. Margarets High School in St. Louis Park Minnesota in 1984, he attended the University of Notre Dame du Lac in South Bend Indiana. He graduated in 1988 with a B.S. in Aerospace Engineering, and as an Reserve Officer Training Corps distinguished graduate he received his regular commission into the United States Air Force in May 1988. Upon entering active duty in January of 1989 he was assigned to the Reentry System Launch Program office of the Ballistic Missile Office at Norton AFB California. During his tour he was responsible for targeting of reentry systems on Developmental Test and Engineering Intercontinental Ballistic Missile (ICBM) launches from Vandenberg AFB California, integration of reentry systems for launch on Minuteman I ICBMs, and planning and coordination of budgets for launches. In May of 1991, he entered the School of Engineering, Air Force Institute of Technology in pursuit of a MS in Astronautical Engineering.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE 15 December 1992	3. REPORT TYPE AND DATES COVERED Master's Thesis		
4. TITLE AND SUBTITLE CANONICAL FLOQUET PERTURBATION THEORY		5. FUNDING NUMBERS		
6. AUTHOR(S) David J Pohlen Captain, USAF		8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GA/ENY/92D-03		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology WPAFB OH 45433-6583		10. SPONSORING / MONITORING AGENCY REPORT NUMBER		
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Not Applicable		11. SUPPLEMENTARY NOTES		
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited		12b. DISTRIBUTION CODE		
13. ABSTRACT (Maximum 200 words) Classical Floquet theory is examined in order to generate a canonical transformation to modal variables for periodic systems. This transformation is considered canonical if the periodic matrix of eigenvectors is symplectic at the initial time. Approaches for symplectic normalization of the eigenvectors had to be examined for each of the different Poincaré eigenvalue cases. Particular attention was required in the degenerate case, which depended on the solution of a generalized eigenvector. Transformation techniques to ensure real modal variables and real periodic eigenvectors were also needed. Periodic trajectories in the restricted three-body case were then evaluated using the canonical Floquet solution. The system used for analyses is the Sun-Jupiter system. This system was especially useful since it contained two of the more difficult Poincaré eigenvalue cases, the degenerate case and the imaginary eigenvalue case. The perturbation solution to the canonical modal variables was examined using both an expansion of the Hamiltonian and using a representation that was considered exact. Both methods compared quite well for small perturbations to the initial condition. As expected, the expansion solution failed first due to truncation after the third order term of the expansion.				
14. SUBJECT TERMS Floquet Theory, Perturbation Theory, Periodic Systems, Canonical Floquet Theory, Canonical Transformations, Modal Variables, Symplectic Normalization			15. NUMBER OF PAGES 160	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	