Technical Report 1055

Multistep Methods for Integrating the Solar System

Panayotis A. Skordos

MIT Artificial Intelligence Laboratory

P22220

This blank page was inserted to preserve pagination.

MULTISTEP METHODS FOR INTEGRATING THE SOLAR SYSTEM

by

Panayotis Avgoustos Skordos

Submitted in partial fulfillment of the requirements
for the degree of Master of Science at the Massachusetts Institute of Technology
Thesis Supervisor: Gerald Jay Sussman, Professor of Electrical Engineering

ABSTRACT

High order multistep methods, run at constant stepsize, are one of the most effective schemes for integrating the Newtonian solar system, for extended periods of time. I have studied the stability and error growth of these methods, when applied to harmonic oscillators and two-body systems like the Sun-Jupiter pair. I have found that the truncation error of multistep methods on two-body systems grows in time like t^2, and the roundoff like $t^{1.5}$, and I have a theory that accounts for this. I have also tried to design better multistep integrators than the traditional Stormer and Cowell methods, and I have found a few interesting ones. A second result of my search for new methods is that I did not find any predictor that is stable on the Sun-Jupiter system, for stepsizes smaller than 108 steps per cycle, whose order of accuracy is greater than 12. For example, Stormer-13 becomes unstable at 108 steps per cycle. This limitation between stability and accuracy seems to be a general property of multistep methods.

This thesis is dedicated to my grandparents
and those dreamy places and times in Greece
where they come from and I come from.
A little world fading away,
full of stories and secrets.
A truly wonderful world,
just like any other.

ACKNOWLEDGEMENTS

Professors Gerald Sussman and Jack Wisdom have helped me considerably in this research. Gerry who has been my advisor and a good friend for many years deserves special thanks. Thanks are also due to Piet Hut and the Institute of Advanced Study where I learned much about orbital mechanics.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology, supported by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-86-K-0180.

Contents

1 INTRODUCTION 6

2 THE EQUATIONS OF MOTION 10

3 SUMMARY OF ERROR FORMULAS 14

4 THE TIME GROWTH OF ERROR 20

5 DESIGNING NEW INTEGRATORS 37

6 THE CLASS OF 3-POINT METHODS 50

7 WAYS OF REDUCING ROUNDOFF 59

8 CONCLUSION AND OPEN PROBLEMS 71

9 APPENDIX 75

List of Figures

1 Error vs. period for different values of eccentricity. 81

2 Error vs. eccentricity for different values of period. 82

3 Error in Position, Energy, and Angle at eccentricity .24, shown during 2 orbits after 16x1024 revolutions, orbital period 4334 days, Stormer-12, stepsize 32 days. 83

4 Error in Position and Energy at eccentricity .4, shown during first 10 revolutions, orbital period 4334 days, Stormer-11, stepsize 32 days. 84

5 Error in Position and Energy for Stormer-Cowell-12 on the Sun-Jupiter system, stepsize 32 days, (quad precision). 85

6 Error in X, Y, and Position on a two-dimensional *circular* oscillator $y'' = -y$. Shown during first 50 revolutions, stepsize .1, initial conditions $[x, y, x', y'] = [1, 0, 0, 1]$, Stormer-9. 86

7 Error in X,Y, and Position on a two-dimensional *eccentric* oscillator $y'' = -y$. Shown during first 50 revolutions, stepsize .1, initial conditions $[x, y, x', y'] = [1, 0, 0, .5]$, Stormer-9. 87

8 Error in Position and Energy vs. time for Stormer-8 on the Sun-Jupiter system, stepsize 32 days. The time axis on the two top graphs runs from 40 to 46 revolutions, and on the bottom two graphs from 1000 to 1006 revolutions. 88

9 The distinction between Radial and Angular Error. 89

10 Error in Position, and Energy for Stormer-10 and S3N5-10, on the Sun-Jupiter system. Time runs from 0 to 4096 revolutions. 90

11 Spider plots for Stormer-12 and S3N5-12. 91

12 Spider plot for Stormer-13. 92

13 Error in Position and Energy for Stormer-12 and S3N5-12 — Double Precision — 16x1024 revolutions. 93

14 Error in Position and Energy for Stormer-12 and S3N5-12 — Double Precision — 200x1024 revolutions. 94

15 Peculiar cancellation in Position error: eccentricity .25, orbital period 4334 days, Stormer-12, stepsize 32 days, Double Precision. . . 95

16 Error in Position and Energy for Stormer-12 and S3N5-12 — Quad Precision — 16x1024 revolutions. 96

17 Error in Position and Energy for Stormer-12 and S3N5-12 — Quad Precision — 64x1024 revolutions. 97

18 Error in Position and Energy for Stormer-12 and S3N5-12 — Quad Precision — 200x1024 revolutions. 98

19 Spider plots for Cowell-13 and H615-4. 99

List of Tables

1 Truncation error formulas as a function of stepsize, orbital period, and total integration time. 15

2 The Error Constants of some multistep methods, for orders of accuracy between 7 and 14. 51

1 INTRODUCTION

I have studied the numerical error of 2nd degree multistep methods in long term integrations of smooth orbits, such as the motion of the outer planets. The focus of my study has been on understanding the stability and error growth of these methods, when applied to harmonic oscillators and two-body systems like the Sun-Jupiter pair. I have found that the truncation error on two-body systems grows in time like t^2, and the roundoff like $t^{1.5}$, and I have a theory that accounts for this. I have also tried to design new multistep methods, that are more accurate than the traditional Stormer Cowell in integrating planetary orbits, and I have found some new interesting methods. On the other hand, I have also come across serious limitations between the order of accuracy, stability, and arithmetic precision needed to represent the coefficients of a method. These are the main topics in my paper. I will start by saying a few words about the context behind my work.

The reason why scientists are interested in integrating the motion of the planets for hundreds of million years, is a question that was first asked, probably, by Newton himself, when he conceived of his laws of motion and his law of universal gravitation. The question is whether the solar system is stable or not. Although it certainly looks very stable during short time scales, we do not know if it is stable during long time scales. For example, the orbit of Pluto is presently locked in a 2 to 3 resonance with the orbit of Neptune, and that's what prevents close encounters between these two planets, whose orbits cross each other. However, we do not know if Pluto will remain locked in this resonance forever. After a thousand million years, the Sun will be alive and burning like today, but it is questionable whether Pluto will still be locked in a stable orbit.

The study of long term stability of the solar system has been traditionally done using a simplified model of the planets and the Sun. The planets and the Sun are considered ideal point masses, obeying Newton's second law of motion

$F = ma$, and the only force acting on each body is the sum of the gravitational forces between that body and every other one. Secondary phenomena such as tidal friction among the planets, and mass loss of the Sun due to radiation, are usually omitted from the model in order to simplify things. The simplified model results in a set of nonlinear differential equations (Newton's equations), that describe the evolution of the solar system.

In the Newtonian model of the solar system, one assigns to each body, planets and Sun, initial positions and velocities, and lets the system evolve according to Newton's equations. The goal is to solve the equations, to predict the state of the solar system as far into the future as possible. On the other hand, the Newtonian equations for more than two bodies can not be solved analytically, in general, and the solar system is not one of the special cases for which solutions have been found. A natural thing to do is to solve the equations by numerical integration. [1]

Numerical integration of the solar system has been very fruitful during the last decade, and has added much to our knowledge of solar system dynamics. But like every method of investigation, it has its problems. Two major problems are the computational cost and the growth of numerical error. Both problems are related to the total integration time being very large. Computers get tied up for months, in order to integrate the solar system for a few billion years. To minimize the running time, we must use efficient integration methods, and large stepsizes. At the same time, we must not compromise accuracy. When the integration time-scale is one *billion* years, it is very hard to keep the accumulated numerical error small. These problems have motivated my study of integration methods. I have chosen to look at multistep methods because they are very efficient, very accurate, and very simple.

[1] A brief history of solar system integrations can be found in G. Sussman, "The Outer Solar System for 200 Million Years"

My first concern has been to understand the error growth of multistep methods, on smooth two-body systems (low eccentricity) like the Sun-Jupiter pair. I have found that the error grows in time like t^2. This is true for all multistep methods of all orders of accuracy, except the higher the order is, the longer we must wait before the error enters the t^2 region. This is because the truncation error of high order methods is initially very small and is dominated by roundoff, which grows like $t^{1.5}$. But since truncation grows like t^2, it always overtakes roundoff in the long run. In the first part of my paper, I present my study of the error. I discuss a theory for the time growth of the error, and many useful formulas and graphs describing the dependence of truncation error on the stepsize, orbital period, and eccentricity.

The second part of my paper concerns the design of new multistep methods. I describe an algorithmic procedure for deriving the coefficients of 2nd degree multistep methods, and I also discuss the stability of multistep methods on the harmonic oscillator, and two-body systems. These discussions are geared towards practical matters, and the goal is to facilitate the search for new methods. These discussions are the theory behind two programs I have written, to help me search for new methods. The first program computes automatically the coefficients of a method, given a few parameters, and the second one determines approximately the maximum stepsize for which a method remains stable on the harmonic oscillator.

In my search for new methods, I have tried to find methods of the highest possible order of accuracy. My motivation has been the fact that the accumulated truncation error decreases by 2π over the number of steps per cycle, for each increase in the order of accuracy. Despite lengthy searches, I have not found any predictor of order higher than 13, which is stable on the Sun-Jupiter system, for a stepsize of 110 steps per cycle. Moreover, most predictors of order higher than 13 have coefficients that do not fit in 53 bits mantissa (IEEE64). I do not know the reason behind this, but it may be a fundamental limitation of multistep methods,

or polynomial approximation in general.

My search for new methods has led me to a class of three-point methods, which is a generalization of Stormer, and has the following nice property: By varying a parameter, we can trade stability for accuracy. One member of this class, referred to as S3N5, is particularly interesting because it resembles Stormer in its simplicity, but is more accurate than Stormer. I have done extensive comparisons between S3N5 and Stormer by running them on two-body systems of low eccentricity. Section 6 of my paper reports these results.

The experimental data from the comparison between S3N5 and Stormer, serves an additional purpose. It confirms my theory of truncation and roundoff error growth, and it also presents some peculiar examples of the interaction between roundoff and truncation, which at present, I do not understand very well.

At the end of my paper, I have included a section on techniques for reducing roundoff error. These techniques involve quad precision arithmetic, and the summed form transformation. Also, there is an appendix that contains the following items: my program for computing the coefficients of 2nd degree multistep methods, and a listing of the coefficients of the new method, S3N5, which is more accurate than Stormer. I have put together all my figures at the end of the paper, they are listed in the order in which they are referenced.

2 THE EQUATIONS OF MOTION

In this section I review some facts about the Newtonian solar system, which are necessary for the later sections. I describe the differential equations that govern the motion of the planets, and I also discuss a few general things about numerical error.

First, I will consider the motion of an isolated two-body system. The differential equations describing this motion come from simple Newtonian physics. When two point masses are placed near each other in space, they attract one another with a force proportional to the product of their masses and inversely proportional to the square of the distance between them.

$$m_1 x_1'' = -k^2 m_1 m_2 \frac{x_1 - x_2}{r^3} \tag{1}$$

Equation 1 gives the x component of the force acting on m_1 due to m_2 where k^2 is the constant of proportionality. Similar expressions to Equation 1 hold for the y and z components of the force acting on m_1, and accordingly for the force acting on m_2 giving a total of six equations. k is called the Gaussian constant and has the value .01720209895 when the unit of distance is 1 A.U. (Astronomical Unit), the unit of mass 1 $M_\odot$ (Solar Mass), and the unit of time 1 day. These units are commonly used in integrations of the solar system, and I will use them in reporting all the experimental results in later sections.

The six differential equations outlined above are all one needs to describe the motion of a two-body system, and they can be solved either numerically or analytically. However, if we add a third mass m_3 to our system, we have a three-body problem, whose solution can not be found analytically, in general. Fortunately, from a numerical point of view, solving for three bodies is no harder than solving for two bodies, it is only more costly. For three bodies we need 9 force equations, and we also need to add an additional term to each equation — the gravitational

forces obey the principle of superposition. Equation 1 becomes

$$m_1 x_1'' = -k^2 m_1 (m_2 \frac{x_1 - x_2}{r_{12}^3} + m_3 \frac{x_1 - x_3}{r_{13}^3})$$

If we consider the whole solar system consisting of n planets and the Sun, we have $n+1$ bodies, hence $3(n+1)$ force equations with n terms in each equation. These $3(n+1)$ equations can be integrated numerically to predict the future positions of the planets. [2]

One thing to notice about the equations of motion of the solar system is that they are 2nd degree ordinary differential equations involving only the position variables. Therefore, if we don't need the velocities at every step, it is a good idea to use an integration method that makes no use of first derivatives. Otherwise, we have to replace each 2nd degree equation with two 1st degree equations, and this is very costly. The multistep methods I have studied, are designed to make no use of first derivatives; for this reason they are called 2nd-degree methods as opposed to 1st-degree methods like Runge-Kutta and Adams-Moulton.

A second thing to notice about the solar system is that the mass of the Sun is very large compared to all the others. Jupiter has the second largest mass in the system, and its mass is only $\frac{1}{1047} M_\odot$ (solar masses). Because of this, each Sun-planet pair behaves as a two-body system under small perturbations from the rest of the planets. It is reasonable, then, to expect that most of the numerical goods and bads of the solar system also appear in smooth two-body problems

[2] There are other formulations in addition to the Cartesian equations (as they are called) that I outlined above; for example the perturbing-elements formulation and Encke's formulation (a hybrid between elements and Cartesian). The perturbing-elements formulation is often used for short term prediction of the position of the planets (a time-scale of a couple of years). For longer periods of time this method becomes inaccurate, and Encke's formulation is an attempt to salvage the method. I am not aware, though, of any successful integrations using Encke's method in the time-scale of a few hundred million years. I suspect the problem is that Encke's method requires many trigonometric function evaluations, which are very costly and a great source of numerical error.

(low eccentricity like the planets). If a method performs well on smooth two-body problems, it should also perform well on the whole solar system. [3] This is fortunate because it is much easier to test an integration method on the two-body problem than on the whole solar system; the computational cost is smaller, and the numerical error can be computed by comparing the numerical solution against the analytic solution.

In my numerical experiments described in this paper, I have used almost exclusively two-dimensional two-body systems. To measure the error, I have used a simple program that computes the analytic solution in the following way: it reduces the two-body problem to the motion of the difference vector, converts from rectangular to orbital elements, advances in time by solving Kepler's equation, and finally transforms back to the original coordinates. [4]

A few results concerning integrations of the outer solar system, to which I refer in later sections, are based on estimating the position error, as follows: The first technique is by measuring the drift in total energy which must be conserved. The second technique is by integrating forwards and backwards for equal lengths of time. Since, the laws of motion are reversible, the final state after integrating forwards and backwards for equal lengths of time, must be identical with the initial conditions.

[3] This is not strictly true. For example, a two-body orbit, under regularization, becomes a linear oscillator of a fixed frequency (see Stiefel & Scheifele 1971). Further, there are numerical methods that integrate exactly (within roundoff) sinewaves of a known frequency (see Stiefel and Bettis 1969). Hence, a method designed for a specific frequency of the regularized two-body problem will do superbly. But this approach can not be used successfully on the whole solar system. This is not because there will be too many fixed frequencies to match in the integration method. Rather, it is because $3n$ force equations become $4n(n-1)+1$ under regularization which is very costly. Moreover, the transformation to and from regularized coordinates is not numerically pleasant.

[4] For a derivation of the solution of the two-body problem, and the relevant formulas see S. McCuskey, *Introduction to Celestial Mechanics*, 1963, or any decent physics textbook.

One more thing to clarify, is the use of the term error in this paper. Unless otherwise indicated, I have used the term "error" to mean the distance in phase space between the computed solution (computed state) and the actual solution. This has two parts, position error and velocity error, but since 2nd degree methods do not use velocities, error refers to position error. For example, in the case of a planar two-body system, this is

$$\sqrt{(x - x^*)^2 + (y - y^*)^2}$$

where x^*, y^* denote the true solution.

3 SUMMARY OF ERROR FORMULAS

The first step in understanding numerical error is to distinguish between roundoff and truncation. The idea is that roundoff error comes from finite computer arithmetic, and truncation error comes from the finite approximation in the integration method. Clearly, each type of error can exist without the presence of the other, and in this sense they are independent. But in practice, when by choice of stepsize and method their sizes become comparable, they interact and combine to produce all kinds of error behavior. Sometimes they combine additively, but more often, they combine in more complicated ways, partially canceling or re-enforcing each other.

A good strategy is to try to understand first the growth of each type of error when it exists by itself, that is when it's so dominant that we can ignore the presence of the other. By suitable choice of stepsize, we can easily make each type of error become the dominant one, large stepsize will bring out the truncation error, and sufficiently small stepsize will bring out the roundoff.

This section is a collection of formulas concerning the numerical error of multistep methods, when applied to two-body systems. Most formulas deal with truncation error because this is better understood and more predictable than roundoff error. Also, most formulas deal with the case of varying at most one integration parameter while holding the other parameters constant. In some cases (e.g. time scaling) it seems that more than one parameter varies at at time, but a moment's thought reveals that this is only apparent, and that those cases can be reduced to cases of varying one parameter. The parameters I have considered are the integration time, the stepsize, the order of accuracy of the method, the orbital period, and the eccentricity of the two-body system being integrated.

The integration time, stepsize, and orbital period are naturally grouped together because they are all related to time. In table 1, I have summarized the formulas that describe how scaling any one of these parameters affects the ac-

cumulated *truncation error*. Each row in the table corresponds to a different integration example. Each example is a variation of the reference integration in row 0. The only parameters varying are the ones shown in the table. The symbols in the formulas should be interpreted as follows:

t is the total integration time,

h is the stepsize,

p is the orbital period,

ϵ is the accumulated truncation error,

c is an arbitrary scaling constant, and

k is the order of accuracy of the multistep method used.

The new parameters for each example (they are accented) are given in terms of the reference integration of row 0. I have marked rows [1,2,3,4] with an arrow because they are more important than the rest. The rest of the formulas, rows [5,6,7] can be derived from them.

					Comments
0.	h	p	t	ϵ	
$\rightarrow$ 1.	$h' = c \cdot h$	$p' = p$	$t' = t$	$\epsilon' = c^k \cdot \epsilon$	local truncation error
$\rightarrow$ 2.	$h' = h$	$p' = c \cdot p$	$t' = t$	$\epsilon' = \ ?$	experimental graph only
$\rightarrow$ 3.	$h' = h$	$p' = p$	$t' = c \cdot t$	$\epsilon' = c^2 \cdot \epsilon$	truncation error growth
$\rightarrow$ 4.	$h' = c \cdot h$	$p' = c \cdot p$	$t' = c \cdot t$	$\epsilon' = c^{2/3} \cdot \epsilon$	$p = 2\pi \frac{a^{3/2}}{k\sqrt{m_1+m_2}}$
5.	$h' = h$	$p' = c \cdot p$	$t' = c \cdot t$	$\epsilon' = c^{2/3} \cdot c^{-k} \cdot \epsilon$	compose 1 and 4
6.	$h' = c \cdot h$	$p' = p$	$t' = c \cdot t$	$\epsilon' = \ ?$	compose 2 and 4
7.	$h' = c \cdot h$	$p' = c \cdot p$	$t' = t$	$\epsilon' = c^{2/3} \cdot c^{-2} \cdot \epsilon$	compose 3 and 4

Table 1: Truncation error formulas as a function of stepsize, orbital period, and total integration time.

First, I will show the validity of formula 4. It says that if we scale all the time parameters of an integration by the same factor c, then the error will scale by $c^{2/3}$. The reason is that scaling all the time parameters leaves the problem unchanged from a physical point of view. Therefore, the error should remain the same. Indeed, the relative error remains the same. Only the absolute error changes. It scales in proportion to the size of the new orbit, which is proportional to the new semimajor axis, which is proportional to 2/3 power of the new orbital period (Kepler's third law). This is the proof. Of course, I have also tested the formula experimentally. I computed the position error for Stormer-8 on two-dimensional two-body systems of eccentricity .05 after 1000 revolutions (truncation dominated the error). I started with an orbital period of 4334 days and a stepsize of 40 days, which I then scaled by factors ranging from 0.1 to 2.0, keeping the total number of revolutions fixed to 1000. Formula 4 turned out to be surprisingly accurate.

On the other hand, formula 1, which is perhaps the first formula one learns in numerical analysis courses, is not so accurate. The c^k can actually vary around $c^{k\pm.5}$, and perhaps more. This formula comes from an approximate formula for the local truncation error, and some unproven assumptions about how errors get added up in the accumulated truncation error. These assumptions are responsible for the variation in $c^{k\pm.5}$. Nevertheless, formula 1 is very useful in practice.

Formula 3 is also an approximate formula, as much as it's usually the case that most integrations are influenced somewhat by roundoff error. The more dominant the truncation error is, the more accurate formula 3 is. Also, the longer the integration time is, that is the larger the constant c is, the more accurate formula 3 is. I will not give the proof of this formula here. Instead, I have devoted a whole section to it, the next section. I should point out that formula 3, just like formula 4, and unlike formula 1, are specific to the two-body problem, and related problems like the solar system. It should be obvious why.

Formulas 5 and 7 are the composition of 1 and 3 with 4. I must hasten to say

here that *only formula 4 can be composed with the other formulas* because it is in essence an algebraic transformation. The other formulas can *not* be composed with each other. To derive formula 5, use formula 1 with a scale factor c^{-1} and then apply the transformation given by formula 4.

The reason why formulas, such as 1 and 3, can not be composed together is that scaling the stepsize is not independent from scaling the time. When both of them are scaled (by the same or a different factor it doesn't make a difference), the error behaves in a completely different way from what the composition of 1 and 3 suggests. [5] The composition of 1 and 3 implies a power law for the error ($\epsilon' = c^{k+2} \cdot \epsilon$), which if it were true, I wouldn't have put a question-mark in row 6 of the table. Similarly, I wouldn't have put a question-mark in row 2, which describes the dependence of truncation error on orbital period. By applying transformation 4 to row 6, I could trivially derive a formula for row 2. Moreover, row 2 would also be a simple power law. But it isn't. I have tested it.

My experimental graph of truncation error vs. orbital period (see figure 1 on page 81) resembles an exponential curve. However, I have not been able to fit the curve very well with a single exponential, and it is also quite far from being a simple power law. A possible explanation is that the accumulated truncation error is *proportional to the size* of the local truncation error, which is proportional to a high derivative of the acceleration, which can be expressed as a sum of powers in semimajor axis and eccentricity. Therefore, the exponentially-looking curve could be a sum of inverse powers in orbital period.

On the other hand, the previous explanation is not very useful in practice, even if true. High derivatives of the acceleration are very complicated expressions. I computed the 4th derivative using Macsyma, and it took about 5 pages to print. The size of the derivative expression grows doubly exponentially with the number

[5] This reminds me of quantum mechanics, and bullets going through a double slit.

of differentiations. The expression should simplify a little, once it is evaluated at a convenient place, like perihelion passage. Yet, it will probably remain too complicated to be of practical use. I think a simple approximate formula is more desirable. Perhaps, a piecewise exponential formula (different exponentials for different ranges of the argument), which can be found experimentally, will be useful.

The experiments I have done concerning the orbital period, have used the Stormer method, at various orders of accuracy. All my tests have shown an error curve of the same character, that resembles an exponential. I have included a plot of the error curve, for Stormer-8, at the end of this paper (see page 81). Following that figure, I have also included a plot of the error as a function of eccentricity (figure 2 on page 82). This curve also looks like an exponential law, but I can not fit it accurately enough with a single exponential. Perhaps, the same idea as in the case of orbital period (time derivative of the acceleration) can help to understand the dependency of truncation error on eccentricity.

With regard to figures 1 and 2, a few more explanations are necessary. Figure 1 shows a number of position error curves vs. orbital period for different values of eccentricity. Analogously, figure 2 shows position error curves vs. eccentricity for different values of orbital period. I have used the same measurements to create both of these plots, but plotted them differently in each case. I performed the integrations using Stormer-8 at a stepsize of 32 days and a fixed integration time of 4438475.8 days (12160.208 years) or approximately 1024 Jovian revolutions. All the integrations are two-dimensional, in center of mass coordinates, and start at perihelion passage.

THE FORMULAS FOR ROUNDOFF ERROR

When the roundoff error is the dominant source of error, most of the previous formulas need to be changed. An exception may be formula 4, the case of scaling all time parameters, because it is in essence an algebraic transformation. If the whole physical orbit is scaled up, then the roundoff error (in position) should also scale accordingly, just like the absolute truncation error. I have not done any tests, though, to verify the formula. It should be tested.

With regard to the dependence of roundoff error on the total integration time, the new formula is $\epsilon' = c^{1.5} \cdot \epsilon$ where $t' = c \cdot t$. Just like the corresponding formula for truncation error, this formula is the topic of the next section.

Finally, with regard to the dependence of roundoff error on the stepsize, I can only report some observations. As the stepsize decreases, there comes a point where truncation error becomes comparable to roundoff, assuming the total integration time is fixed. Then, there is a region of values of the stepsize, whose size depends on the order of accuracy of the method, where the stepsize seems irrelevant to the accumulated error. The error curve for this range of stepsizes looks approximately flat with noisy ups and downs. The size of this region becomes larger as the order of accuracy of the method increases.

After the flat and noisy region, there comes another region, where the error starts increasing with decreasing stepsize. To my surprise, in the few cases I have looked at, the error grows approximately quadratically with decreasing stepsize. I have seen this both for Runge-Kutta-4 and for Forward Euler, on the two-body problem, and on the decaying exponential problem. So perhaps, there is something universal behind this.

4 THE TIME GROWTH OF ERROR

In this section I discuss the error growth as a function of time, for multistep methods applied to harmonic oscillators and smooth two-body problems. As far as the solar system is concerned, my experiments show that the growth of error on the solar system is similar to the growth of error on smooth two-body systems. I think the error mechanisms in these two cases are similar because the solar system is a collection of smooth two-body systems (Sun-planet pairs). Moreover, the Sun-Jupiter pair is so much more massive than the others, that it dominates the error of the whole system. Therefore, although my error analysis focuses on two-body systems, my results should be useful in integrations of the solar system, also.

I have found that the truncation error of multistep methods on smooth two-body systems grows independently (that is when it dominates roundoff error) as t^2, while the roundoff error grows independently as $t^{1.5}$. I have verified this for Stormer predictors of all orders (including the summed form Stormer formula), for Stormer-Cowell predictor-correctors, for some new predictor methods I have designed, and for Runge-Kutta-4. I should add that, in the past, there have been papers claiming quite the opposite of what I just stated. They have claimed that Runge-Kutta-4 and 1st degree multistep methods (of even order) have linear error growth on two-body systems. [6] I haven't been able to understand the elaborate proofs of these reports, and the experimental data is minimal and obscure.

One thing I should make clear in this discussion, is that the t^2 truncation error refers to position error. The energy error grows at a power of t smaller than the position error. It grows like t, linearly with time. The reason will become clear when I discuss later in this section, how *the shear flow* of a two-body orbit transforms a linear error in semimajor axis (the energy error is proportional to

[6] P. Henrici *Error Propagation for Difference Methods*, page 55, 1963. Henrici's results are also quoted by E. Stiefel, *Celestial Mechanics: 2*, page 274.

the error in semimajor axis) to a quadratic error in position.

A second thing to clarify is the following: In trying to measure the truncation error experimentally, sometimes, it takes a long time before the truncation error develops and overtakes roundoff. How long it takes, depends on the order of accuracy of the method, the stepsize used, and the orbital period and eccentricity of the two-body system being integrated. Low order of accuracy, large stepsize, small period, and large eccentricity increase the local truncation error. The larger the local truncation error is, the sooner the accumulated error exhibits a clear t^2 error growth. On the other hand, the smaller the local truncation error is, the longer it takes before truncation grows large enough to dominate roundoff. However, it always grows larger.

An important thing to note is that I am talking about error growths over large time scales. On a small time scale, say during one orbit of a two-body system of eccentricity .05 and period 4334 days (the Sun-Jupiter system), one can not observe the t^2 truncation growth, nor can one observe the $t^{1.5}$ roundoff growth. On a small time scale, the error (in position) oscillates like a sinusoid. In order to see the t^2 growth, one must measure the error over hundreds or even thousands of revolutions, depending on the problem. Examples are given in my figures at the end of the paper.

With regard to measuring the growth in energy error, which is linear when truncation dominates, it is a good idea to measure the error at aphelion passage and not at perihelion. If the measurements are taken, say every 100 revolutions for a total of 100,000 revolutions, then this may not matter too much. But if the measurements are taken once every revolution at perihelion passage (for a problem of eccentricity .25), then the resulting error curve will have a lot of noise to it, which may be confusing. The noise is an artifact of sampling the perihelion region where the energy error changes very abruptly. This becomes all the more true, as the eccentricity increases. For an illustration at eccentricity .24 see figure 3 on

page 83.

If the eccentricity becomes too large, say .5, then the theory I am advancing in this paper is not very applicable any more. This should not raise doubts about the theory. The same dynamics of shear flow and resonances in truncation error seem to take place for high eccentricities, also. The problem is, however, that for high eccentricities the position error oscillates so greatly during the course of a single orbit (its amplitude is comparable to the accumulated error), that it's not possible to measure an overall growth of error as in lower eccentricities. Figure 84 on page 84 illustrates the difficulty. It shows the error during the first 10 orbits of a two-body system of eccentricity .4 and period 4334 days, for Stormer-11 at a stepsize of 32 days.

On the other hand, figure 5 on page 85 shows the error on a two-body system of .05 eccentricity. The method is Stormer-Cowell-12 at 32 days stepsize, using quad precision arithmetic, which reduces the roundoff error (I explain this in Section 7). Figure 5 shows two different stages of the integration: first after a few revolutions, and secondly after 256x1024 revolutions. At the beginning the error is purely roundoff. After 256x1024 revolutions (which took a couple of days to compute) the error is dominated by truncation and grows like t^2.

The abscissa (time) on the top graph runs from 0 to 4096 revolutions (Jovian years), and the one on the bottom runs from 0 to 256x1024 revolutions, which is about 3 million years. The two-body system integrated has masses and orbital elements corresponding to Sun and Jupiter as given by CHO [7], but converted to two-dimensions on the plane of orbit. The conversion results in the following initial positions and velocities (center of mass coordinates):

[7] C.Cohen, E.Hubbard, and C.Oesterwinter: "Elements of the outer planets for one million years." *Astronomical Papers of the American Ephemeris*, XXII, I, 1973.

	x	y	x'	y'
Sun	-4.720912507067483e-3	0	0	-7.55789984986842e-6
Jupiter	4.944500871054731	0	0	7.915851508595781e-3

The 13 initial states needed to start the integration (12th order method) were generated by solving the two-body equations analytically. The position error shown in figure 5, is Jupiter's computed position minus Jupiter's exact position (calculated analytically). The energy error shown in the same figure, is the computed total energy minus the initial total energy, which has the value -2.7144e-8 in the usual astronomical units. [8] I have used the same conventions for measuring errors in all my examples.

[8] See page 10 for an explanation of units.

THE THEORY OF THE ELLIPTICAL RACING TRACK

Before getting into the details of the mechanism that causes the t^2 and $t^{1.5}$ error growths, I think it is interesting to visualize the dynamics of a two-body orbit in terms of the following scenarios.

Consider an ellipse of small eccentricity, say .05 like Jupiter's orbit. Imagine you lay down the ellipse on a table, and suppose that suddenly, the ellipse becomes a racing track. There are two cars sitting on the track, waiting for the start signal to go. Car-A is slightly further in front of car-B. Let's call this small distance between A and B, the "error". Now, consider what happens to the error as the cars start spinning around the elliptical track. The cars move like two independent Jupiters orbiting an imaginary Sun sitting at one of the foci of the ellipse. What happens is the error oscillates like a sinusoid, it grows and shrinks in size. It becomes largest when car-A is at perihelion passage, and smallest when car-A is at aphelion passage. Once the race has started this way, the error will continue to oscillate forever. This is the first scenario.

The second scenario deals with a slightly different situation. Now, we have two elliptical racing tracks, and one of them is slightly, but only so slightly, larger than the other. Viewed as mathematical ellipses, the racing tracks have the same eccentricity, and their foci coincide. There are two cars again, one on each elliptical track. Car-A is on the smaller elliptical track. As the start signal goes off, the cars begin to race. The question is which one is going to win. The answer is car-A is going to win because its track is smaller. The interesting thing, though, is that the error grows linearly with time. Further, if we look at it closely, say during the course of one revolution, we see an oscillation on top of the linear curve. This oscillation is of the same kind as the oscillation in the first scenario. Incidentally, higher eccentricity racing tracks have steeper oscillations in the error and of wider amplitude.

The third scenario is a small modification of the second one. Now car-A is endowed with an unusual elliptical track, quite different than any tracks one is likely to see in real life. This track grows in size. It grows very, very slowly, so slowly that it takes dozens of revolutions before we can observe any change. Yet, it grows, and it grows linearly with time. Now the question is what happens to the error between car-A and car-B. The error grows quadratically in time. It always grows one power of t higher than the growth of the growing racing track.

I think the most interesting points about the preceding scenarios are two things: the oscillations of the error, and the fact that the error grows in time one power of t higher than the growth of the ellipse. The theory I am proposing says, basically, that there is a resonance in the truncation error of the two-body problem, which makes the ellipse (the semimajor axis) grow linearly in time. Then, the shear flow of the two-body system integrates the linear growth one power of t higher. In the rest of the section, I first discuss the resonance. I start by looking at the error of multistep methods on the harmonic oscillator. A similar truncation resonance exists for the harmonic oscillator, but it is simpler than the one of the two-body problem.

ERROR ACCUMULATION ON THE HARMONIC OSCILLATOR

$$y'' = f(y) = -\omega^2 y$$

My numerical experiments show that the truncation error on the harmonic oscillator grows linearly with time, and that the roundoff error grows like $\sqrt{t}$. I have performed these tests both on one-dimensional and two-dimensional oscillators. The reason for looking at two-dimensional harmonic oscillators is that they correspond more closely to two-body orbits. Under appropriate initial conditions a system of two harmonic oscillators of the same frequency performs an elliptical

trajectory on the plane like a two-body system does, except the linear trajectory has constant angular velocity which is independent of radius while the two-body does not. [9]

The $\sqrt{t}$ growth in roundoff error is a natural thing to expect, it is the average distance that a drunken man walks, with his eyes closed. On the other hand, the linear growth in truncation error is caused by a resonance between the local truncation error and the homogeneous part of the error equation. This resonance is the key idea. I will explain it in a little more detail. The difference equation that computes the solution, that is the equation that comes from the integration method, looks like the following formula which is actually Stormer-12 when the b_i are chosen appropriately.

$$y_{n+1} = 2y_n - y_{n-1} + h^2 \left[b_0 f(y_n) + b_1 f(y_{n-1}) + \cdots + b_{12} f(y_{n-12})\right]$$

I picked this formula in order to have a concrete example. Now if $z_n = z(nh)$ denotes the true solution, then the local truncation error d_n can be defined by

$$z_{n+1} = 2z_n - z_{n-1} + h^2 \left[b_0 f(z_n) + b_1 f(z_{n-1}) + \cdots + b_{12} f(z_{n-12})\right] - d_n$$

This comes from substituting z_n in the previous equation for y_n; any discrepancy is the local truncation error. If we let $\epsilon_n = y_n - z_n$ denote the accumulated truncation error, and subtract the two previous equations we get an equation for ϵ_n

$$\epsilon_{n+1} = 2\epsilon_n - \epsilon_{n-1} + h^2 \left[b_0(f(y_n) - f(z_n)) + \cdots + b_{12}(f(y_{n-12}) - f(z_{n-12}))\right] + d_n$$

We can simplify this by substituting $f(y) = -\omega^2 y$. We get finally the error equation

$$\epsilon_{n+1} = 2\epsilon_n - \epsilon_{n-1} - (h\omega)^2 [b_0\epsilon_n + b_1\epsilon_{n-1} + \cdots + b_{12}\epsilon_{n-12}] + d_n$$

[9] Another difference, which is not relevant to this discussion, is that the center of the linear ellipse coincides with the xy-origin. For the two-body, the xy-origin is one of the two foci of the ellipse.

which looks exactly like the equation for y_n (the numerical solution), except there is an additional drive term d_n. Therefore, the homogeneous part of the error equation must have a principal root (a complex conjugate pair) that approximates the true solution, namely $cos(\omega t)$. All other roots must lead to decaying solutions if the method is stable.

It remains now to estimate the drive term d_n which is the local truncation error. We recall the approximate formula

$$d_n \simeq C \cdot h^{13} \cdot f^{(13)}(\xi_n)$$

where C is the method's error constant, h the stepsize, and $f^{(13)}(\xi_n)$ is something like $\omega^{13} sin(\omega \xi_n)$ for the harmonic oscillator. Hence, the drive term of the error equation oscillates at the same frequency as the homogeneous part, creating a resonance, a linear error growth

$$\epsilon_n \sim n cos(\omega n)$$

This is exactly what experimental tests indicate. The truncation error on the one-dimensional harmonic oscillator grows like $tsin(t)$. Similarly the truncation error on the two-dimensional harmonic oscillator grows like $tsin(t)$ in the x and y coordinates, and like $t(A + Bsin(t))$ in position. This is because the errors in x and y are approximately 90 degrees off phase, that is

$$\begin{aligned} \Delta x &= C_1 \cdot tsin(t) \\ \Delta y &= C_2 \cdot tcos(t + \phi) \end{aligned}$$

where C_1 C_2 are constant scale factors and ϕ is a small constant. The position error is given by $\sqrt{(\Delta x)^2 + (\Delta y)^2}$. Note that if Δx and Δy were exactly 90 degrees off phase, *and* if their corresponding scale factors were exactly equal, then the position error would be a perfect t, and the factor B in the position error formula $t(A + Bsin(t))$ would be zero. But in practice the factor B, which describes the wiggliness of the linear growth, is never zero but varies according to the problem.

For example, on eccentric (as opposed to circular) two-dimensional harmonic oscillators, which arise from initial conditions such as $(x, y, x', y') = (1, 0, 0, .5)$, the wiggliness B is large because the corresponding scale factors in Δx and Δy differ by a large amount. See figures 6 and 7 starting on page 86 for an illustration of this.

ERROR ACCUMULATION ON THE TWO-BODY PROBLEM

I will now examine the error growth of multistep methods when applied to two-body systems of low eccentricity. I am proposing that a similar resonance as the one for the harmonic oscillator, exists here, also. This resonance gives rise to a linear error growth, which becomes quadratic in the angular direction because "different radia travel at different speeds". I will start by discussing how the linear resonance develops.

To derive the error equation, I subtract the difference equations for y_{n+1} and z_{n+1} just like I did in the case of the harmonic oscillator. To simplify the term $f(y_n) - f(z_n)$ I use the Mean Value Theorem

$$f(y_n) - f(z_n) = \left[\frac{\partial f}{\partial y}\right]_n (y_n - z_n) = \left[\frac{\partial f}{\partial y}\right]_n \epsilon_n$$

The equations for the error and the computed solution look as follows

$$\begin{aligned} \epsilon_{n+1} &= 2\epsilon_n - \epsilon_{n-1} + h^2 \left[b_0 \left[\tfrac{\partial f}{\partial y}\right]_n \epsilon_n + \cdots + b_{12} \left[\tfrac{\partial f}{\partial y}\right]_{n-12} \epsilon_{n-12} \right] + d_n \\ y_{n+1} &= 2y_n - y_{n-1} + h^2 \left[b_0 f(y_n) \quad + \cdots + b_{12} f(y_{n-12}) \right] \end{aligned}$$

The goal is to show that the homogeneous parts of these two equations have approximately the same periodic solution, and that the drive term d_n in the error equation resonates with this solution.

For very small eccentricities (approaching zero) it's easy to see why this should be true. As the eccentricity approaches zero, the two-orbit becomes more and more like a harmonic oscillator, spinning around a constant radius with constant angular velocity. The acceleration looks more and more like

$$f(y_n) \simeq \left[\frac{\partial f}{\partial y}\right]_n \cdot y_n$$

and the error drive varies like a cosine of the same frequency as the solution. A resonance between the drive and the "feedback" in the error (the homogeneous part of the equation) takes place, which causes a linear growth in error, just like

the harmonic oscillator. In the next subsection I will explain how this linear error becomes quadratic. It suffices now, to say that although for zero eccentricity, the two-body orbit looks very much like a harmonic oscillator, the two-body never loses its nonlinear character. The nonlinearity causes *angular shear* which integrates the linear error into a quadratic error in angle.

When the eccentricity of the two-body problem becomes significantly greater than zero (say .4), the situation is not as simple as for eccentricity close to zero. I have not been able to prove formally the existence of a resonance for higher eccentricities. Yet, the experimental data shows that resonances exist for higher eccentricities, also. Perhaps an argument might proceed along the following lines:

For the harmonic oscillator the local truncation error is a sinusoid at the frequency of the solution. For the two-body it is more like a combination of sinusoids of related frequencies that vary with time. To see this, we can write $x = rcos\theta$ where θ is the angle (true-anomaly), and differentiate twice to get the acceleration:

$$x'' = r''cos\theta - 2r'\theta'sin\theta - r(\theta')^2cos\theta - r\theta''sin\theta$$

If we repeat differentiating, we get a sum of terms in $\theta', (\theta')^2, (\theta')^3, \ldots$ as well as terms in $\theta'', \theta''', \theta^{iv}, \ldots$ and so on. Therefore, the local truncation error, which is proportional to a high derivative of the acceleration, is a combination of these frequencies. Although these frequencies vary with time, they do not vary too much, and their changes are periodic as we trace out a complete orbit. The same frequencies (at least the lower ones) are also found in the true solution, and the lowest frequency, which corresponds to one oscillation per orbit, is probably the most dominant of all. So, it may be reasonable to think that if the local error drive, d_n, appeared as a drive term in the *difference equation for the solution*, we would get a nice resonance. Such an equation would look like this:

$$y_{n+1} = 2y_n - y_{n-1} + h^2 \left[b_0 f(y_n) + b_1 f(y_{n-1}) + \cdots + b_{12} f(y_{n-12})\right] + d_n$$

The question is why d_n resonates with the error equation whose terms are different from those above. But perhaps they are not so different. The terms to compare are the following:

$$f(\epsilon_n) \quad \text{and} \quad \frac{\partial f}{\partial y}\epsilon_n$$

We want these to be approximately the same. If we expand in Taylor Series

$$f(\epsilon_n) = f(0) + \frac{\partial f}{\partial y}\epsilon_n + \ldots$$

and observe that ϵ_n is quite small, so we can neglect high order terms, and that $f(0)$ is zero because, for example, in the case of a two-dimensional reduced-mass two-body we have

$$f(0) \;= -\frac{\epsilon_x}{(\epsilon_x + \epsilon_y)^{3/2}}\bigg|_{\epsilon_x=0} = 0$$

then we can see that the homogeneous error equation must have the same behavior as the computed solution. In this way, we get a resonance.

As I said at the beginning of this discussion, the preceding argument is not intended to be a complete proof. On the other hand, I think I have made a good case for the truncation resonance when the eccentricity approaches zero. So, in the next subsection I will proceed to discuss the shear flow of two-body orbits, which is very well understood, and explains how a linear error with a radial component gets integrated into a quadratic error.

The last item in this subsection is experimental evidence for the error resonance. Figure 8 on page 88 shows the energy and position errors for the Sun-Jupiter system during 6 consecutive revolutions. The top graphs show the energy and position errors after 40 revolutions. The blips in energy occur at perihelion passage. On the two top graphs we can see a major resonance at one cycle per orbit, but there is also a secondary oscillation (mostly in the position error) at half the fundamental frequency. Perhaps there are other frequencies as well that

can not be resolved here, and perhaps roundoff also plays a role during the initial parts of the integration.

On the other hand, if we wait long enough, the initial transients die away. The two bottom graphs show the error after 1000 revolutions. Now the major resonance that follows the solution has outgrown all initial transients. The position error oscillates once per orbit and peaks at perihelion passage. We can also see an upward drift in the position error as the error accumulates, and a downward drift in total energy as energy is lost. The integration method is Stormer-8 at 32 days stepsize, and the Sun-Jupiter system is a planar version of CHO elements.

SHEAR FLOW

Having seen that a truncation resonance exists for the two-body problem which is very much like the harmonic oscillator, I will now explain in detail how the linearly growing error becomes a quadratic error in position.

The key idea is a property of nonlinear orbits (such as two-body orbits) which is called *angular shear*. This is loosely speaking the property that different radia go at different frequencies, and it is the cause of Liapunov instability. A system is Liapunov stable if any two trajectory points that are initially close to each other, remain close to each other forever. Clearly this is not the case for the two-body problem because we can consider two elliptical orbits that are arbitrarily close to each other, but one of them has a slightly larger semimajor axis than the other. We can then choose a point on each orbit, so that the two points will be arbitrarily close to each other initially. Yet, the two points will separate from each other as time progresses because their corresponding periods of revolution will be different. The period of revolution of a two-body system is given by

$$P = 2\pi \frac{a^{3/2}}{k\sqrt{m_1 + m_2}}$$

where a is the semimajor axis.

From a numerical point of view, the dependence of the period on the size of the elliptical orbit means the following: If we have a perfect integration except for an initial error in position, which has a radial component and causes an error in semimajor axis, then the error in angle will grow linearly with time. If there are more radial errors in position, that occur throughout the course of the orbit, then these errors will be compounded.

The way an error in position causes an error in semimajor axis a can be seen from the relation

$$a = \frac{r}{1 - e\cos E}$$

where E is the eccentric anomaly and e the eccentricity of the orbit. In particular, if the resonant error in coordinates x and y has a component in the radial direction, then we will get a proportional error in semimajor axis - the error in eccentricity is too small to even out the change $r + \Delta r$ and to leave the fraction unchanged. Further, if the resonant error in x and y grows like t, then the error in semimajor axis will also grow like t because the resonant error does not distinguish direction and always has a radial component. In contrast to this, the error caused by shear flow does not have a radial component, it is only angular and can not affect the semimajor axis.

The idea is to observe that the position error is caused by two mechanisms. The first mechanism is the truncation resonance which causes a linear error growth in position in all directions. Therefore, it also causes a linear error growth in semimajor axis. The second mechanism is the shear flow which is triggered by the semimajor axis error and causes a quadratic error in position. The quadratic error in position takes place in the angular direction only, along the orbit, and does not contribute anything to the semimajor axis error. In this way the two errors, linear and quadratic, can coexist and not interfere with each other. The only mixing of errors occurs for the angular component of the resonance which is added to, and overshadowed by the quadratic shear flow.

Figure 9 on page 89 explains graphically how the two error mechanisms coexist. In that figure imagine that m_1 is centered at the origin and m_2 is orbiting around it. The position marked z corresponds to the true solution, y corresponds to the computed solution, θ and ϕ are the corresponding angles measured from perihelion passage. Further, consider z as a function of angle $z = F(\phi)$. The idea is that although the position error (the distance from y to z) grows quadratically

$$|y - z| = |y - F(\phi)| \sim t^2$$

if we subtract from y the angular-shear, or equivalently if we consider the true

solution at the angle of the computed solution we see a linear drift.

$$|y - F(\theta)| \sim t$$

The $|y - F(\theta)|$ error is only a consequence of the truncation resonance. The $|z - F(\theta)|$ error is mostly a consequence of shear flow.

All that remains to finish the argument, is to show how a linear error in semi-major axis causes a quadratic error in angle and hence position. For this, it suffices to prove that a linear error in semimajor axis causes a quadratic error in mean anomaly M. This is by virtue of the relation

$$M = E - esinE = (1 - e)E + e\frac{E^3}{6} + \cdots$$

where E is the eccentric anomaly. The previous relation shows that a quadratic drift in M causes a quadratic drift in E. Further, if E drifts quadratically, then for geometrical reasons both the true anomaly θ and the position also, drift quadratically. Now, to see why a linear error in semimajor axis a, causes a quadratic error in mean anomaly M, consider the relation

$$M = \eta \cdot t \qquad \text{where} \qquad \eta^2 = \frac{k^2(m_1 + m_2)}{a^3}$$

where η is the mean-daily-motion. By differentiation

$$\Delta\eta = -\frac{3\eta}{2a}\Delta a$$

and hence for small changes (such as numerical errors) $\Delta\eta \sim \Delta a$, and if $\Delta a \sim t$ then $\Delta\eta \sim t$, and finally

$$\Delta M = \Delta\eta \cdot t \sim t^2$$

and we are done.

I have thus explained how the t^2 truncation error comes into being. It is now trivial to account for the $t^{1.5}$ roundoff error also; the reader should be reminded that my

analysis of the error treats only the case where each type of error exists by itself. We saw that the roundoff error on the harmonic oscillator grows independently like $\sqrt{t}$ because computer roundoff is a random walk process. Similarly, on the two-body problem roundoff has a $\sqrt{t}$ growing component, but like the truncation resonance the $\sqrt{t}$ error affects the semimajor axis error, and triggers shear flow. Similarly to above, we get

$$\Delta\eta \sim \Delta a \sim \sqrt{t}$$

and

$$\Delta M = \Delta\eta \cdot t \sim t^{1.5}$$

In other words, the roundoff process produces a radial error just like truncation does, but the shear integrates both of them a power of t higher. That's the whole idea. This finishes my analysis of the error as a function of time.

5 DESIGNING NEW INTEGRATORS

The purpose of this section is to discuss the design of new multistep methods for 2nd degree differential equations. The main points are an algorithm for computing the coefficients of predictor and corrector methods, and a discussion concerning stability and order of accuracy. I will start by explaining why I chose to study exclusively multistep methods.

The main advantages of constant-stepsize multistep integrators are their simplicity, efficiency, and accuracy. The programs that implement such methods are typically short and simple to execute (therefore fast) because they do not involve any complicated control decisions during the computation. Also, the simplicity of multisteps methods makes it easy to build a special purpose computer to execute them, and to run long integrations at low cost. An example of such a computer is the *Digital Orrery*, which is a very simple machine and runs 60 times faster than a VAX 11-780, in integrating the solar system using multistep methods. Presently, the Digital Orrery holds the record for the longest integrations of the solar system ever.

With regard to accuracy, constant-stepsize multistep methods are very good when the problem to be integrated is smooth, such as the motion of planets. They remain accurate even for large stepsizes, if the order of accuracy (the number of states used in every integration step minus one) is big enough. This property of multistep methods is very important for long term integrations because large stepsizes reduce the roundoff error and the total running time. Efficiency is of primary concern in integrations of the solar system which may require months to compute, using present day technology.

The concern for efficiency in solar system integrations further suggests the use of predictor-only multistep methods. This is because each correction step in a predictor-corrector scheme requires recomputing all the forces between every

pair of bodies. Force computation is the most costly step in the integration. Most integrations of the solar system I am aware of, have used predictor-only methods. [10] For this reason my focus in designing new methods has been on predictors. In this section, however, I discuss both the derivation of predictor and corrector methods.

THE COEFFICIENTS FOR PREDICTORS

I will start by writing down the general form of 2nd degree predictor multistep methods. If the differential equation (or system of equations) is of the form $y'' = f(y)$, then the predictor formula looks as follows:

$$y_{n+1} = a_0 y_n + a_1 y_{n-1} + \cdots + h^2 \left[b_0 f(y_n) + b_1 f(y_{n-1}) + \cdots + b_k f(y_{n-k}) \right] \qquad (2)$$

I will consider the a_i as parameters which determine the method, and the b_i as coefficients to be derived once the a_i are chosen. The b_i are derived so as to make formula 2 exact for all polynomials up to degree k. The a_i are chosen freely except for one requirement, that they satisfy $y(t) = 1$ and $y(t) = t$ because in these two cases $y'' = f(y) = 0$, and the b_i can not be used to make the formula exact. There is also a question of stability, but I will get to it later. The purpose of making formula 2 exact for $y(t) = 1, t, \ldots, t^k$ is to match the Taylor Series of the solution locally up to kth order. This is equivalent to making a local polynomial approximation of degree k to the true solution, and using the polynomial to integrate one step further. To derive the coefficients $b_0, \ldots, b_k$ we can substitute in formula 2 the polynomials $y(t) = 1, t, \ldots, t^k$, one by one, and match terms. However, there is a more convenient way which can be formulated so as to be easily programmed on a computer. I will outline this formulation.

[10] For example, Sussman and Wisdom: "The outer solar system for 200 Million years"; Cohen, Hubbard, and Oesterwinter: "Elements of the outer planets for one million years"; Kinoshita and Nakai: "Stability of Pluto's orbit and orbital motions of outer planets for five million years."

We rewrite equation 2 in difference form as follows:

$$y_{n+1} = a_0 y_n + a_1 y_{n-1} + \cdots + h^2 \left[\gamma_0 \Delta^0 f(y_n) + \gamma_1 \Delta^1 f(y_n) + \cdots + \gamma_k \Delta^k f(y_n)\right] \quad (3)$$

where $\Delta^0 f(y_n) = f(y_n)$, $\Delta^1 f(y_n) = f(y_n) - f(y_{n-1})$ and so on.
Then we choose the exponential function

$$y(t) = \frac{h^2}{(logz)^2} z^{t/h} \qquad \text{and} \qquad f(y) = z^{t/h}$$

and substitute $y_n = y(nh) = h^2 z^n/(logz)^2$ and $f(y_n) = z^n$. To determine the coefficients $\gamma_0, \ldots, \gamma_k$ we expand the exponentials z^n, and match terms up to order k (actually it's convenient to expand in terms of another variable ζ to be introduced shortly). This is equivalent to matching individual polynomials one by one up to degree k. Also, note that the b_i are related to the γ_i by the summation

$$\boxed{b_i = (-1)^i \sum_{j=i}^{k} \frac{j!}{(j-i)!\, i!} \gamma_j \; ; \quad i = 0, \ldots, k} \quad (4)$$

where k is the highest difference used in formula 3.

To carry out the matching of the terms, note that $\Delta^1 f(y_n) = z^n(1 - z^{-1})$ and $\Delta^2 f(y_n) = z^n(1 - z^{-1})^2$ and so on. For shorthand, let's also set $\zeta = 1 - z^{-1}$, or equivalently $z = (1 - \zeta)^{-1}$. We get

$$\frac{h^2 z^n}{(logz)^2} \left[z - a_0 - a_1 z^{-1} - \cdots - a_m z^{-m}\right] = h^2 z^n \left[\gamma_0 + \gamma_1 \zeta + \gamma_2 \zeta^2 + \cdots + \gamma_k \zeta^k\right]$$

Further, we multiply both sides by $(logz)^2 = [log(1 - \zeta)]^2$, divide by ζ^2, and let $\rho(z) = z - a_0 - a_1 z^{-1} - \cdots - a_m z^{-m}$. This gives

$$\frac{\rho(z)}{\zeta^2} = \left[\frac{log(1-\zeta)}{\zeta}\right]^2 \left(\gamma_0 + \gamma_1 \zeta + \gamma_2 \zeta^2 + \cdots + \gamma_k \zeta^k\right)$$

or

$$\frac{\rho(z)}{\zeta^2} = \left(1 + \frac{2h_2}{3}\zeta + \frac{2h_3}{4}\zeta^2 + \cdots\right) \left(\gamma_0 + \gamma_1 \zeta + \gamma_2 \zeta^2 + \cdots + \gamma_k \zeta^k\right) \quad (5)$$

where $h_m = 1 + 1/2 + \cdots + 1/m$ is the mth partial sum of the harmonic series and comes from expanding $[log(1 - \zeta)/\zeta]^2$ in terms of ζ.

Now we have expressed the problem in standard form. Given the parameters $a_0, \ldots, a_m$, we expand $\rho(z)$ in terms of ζ, and match terms in the two sides of equation 5. We can write the left side as follows:

$$\frac{\rho(z)}{\zeta^2} = \frac{1}{\zeta^2(1-\zeta)}\left[1 - a_0(1-\zeta) - a_1(1-\zeta)^2 - \cdots - a_m(1-\zeta)^m\right]$$

and combine the binomial terms into a series in ζ. If the a_i were chosen appropriately so as to satisfy $y(t) = 1$ and $y(t) = t$, then we should get an extra factor of ζ^2 in combining the binomials, and this should cancel the ζ^2 in the denominator. Then a long division by $(1-\zeta)$ gives the desired series expansion for the left side of equation 5. Let's denote this series by $\rho_0 + \rho_1\zeta + \rho_2\zeta^2 + \rho_3\zeta^3 + \ldots$

The series expansion for the right side of equation 5 can be computed by series multiplication. However, if we write down the result for the first few terms, we immediately see a nice recursion relation. For example, the coefficient of ζ^0 is γ_0, the coefficient of ζ^1 is $(\frac{2h_2}{3}\gamma_0 + \gamma_1)$, the coefficient of ζ^2 is $(\frac{2h_3}{4}\gamma_0 + \frac{2h_2}{3}\gamma_1 + \gamma_2)$, and so on. By equating coefficients of equal powers of ζ in this expansion and the $\rho_0+\rho_1\zeta+\rho_2\zeta^2+\rho_3\zeta^3+\ldots$ expansion, we arrive eventually at the recursion relation for the coefficients $\gamma_0, \ldots, \gamma_k$.

$$\boxed{\begin{aligned} \gamma_0 &= \rho_0 \\ \gamma_m &= \rho_m - \tfrac{2h_2}{3}\gamma_{m-1} - \tfrac{2h_3}{4}\gamma_{m-2} - \cdots - \tfrac{2h_{m+1}}{m+2}\gamma_0\ ; \quad m = 1, 2\ldots \end{aligned}} \tag{6}$$

The preceding derivation may appear lengthy, but the actual computing steps are simple, and they can be easily programmed in an appropriate language. I used the Scheme programming language [11] which is very suitable for expressing mathematical ideas. A listing of my program appears in the appendix. It is about 2 pages long and should be easy to read. There are two major procedures, `gamma` and `gamma->beta`, which can be used to derive any predictor of any order that follows the general form. For example, to derive the Stormer γ_i coefficients, we

[11] See Abelson and Sussman, Structure and Interpretation of Computer programs.

evaluate the Scheme expression `(gamma '(2 -1))`, and to get the b coefficients, we evaluate the expression `(gamma->beta (gamma '(2 -1)) 13 i)` where `i` runs from 0 to 12.

THE COEFFICIENTS FOR CORRECTORS

As far as corrector methods are concerned, their coefficients can be derived using the same program as the one for predictors. This is because the coefficients γ_m^* of a corrector obey the relations

$$\boxed{\begin{aligned} \gamma_0^* &= \rho_0 \\ \gamma_m^* &= \gamma_m - \gamma_{m-1}\,; \quad m = 1, 2 \ldots \end{aligned}} \tag{7}$$

where γ_m are the difference-form coefficients of a predictor with the same a_i coefficients as the corrector. Further, the same summation that turns the γ_i into b_i for a predictor turns the γ_i^* into b_i^* for a corrector. To verify formula 7 we follow the same approach that we used to derive the coefficients of a predictor method. The formulas and the expansions in ζ are almost the same, except for one minor difference, an extra z factor. This extra z is responsible for formula 7. I will sketch the idea. The general form of a corrector can be written in terms of b_i^* coefficients as follows:

$$y_{n+1} = a_0 y_n + a_1 y_{n-1} + \cdots + h^2 \left[b_0^* f(y_{n+1}) + b_1^* f(y_n) + \cdots + b_k^* f(y_{n+1-k})\right]$$

The corresponding difference form is

$$y_{n+1} = a_0 y_n + a_1 y_{n-1} + \cdots + h^2 \left[\gamma_0^* \Delta^0 f(y_{n+1}) + \gamma_1^* \Delta^1 f(y_{n+1}) + \cdots + \gamma_k^* \Delta^k f(y_{n+1})\right]$$

Note that $\Delta^0 f(y_{n+1}) = f(y_{n+1}) = z^{n+1}$, $\Delta^1 f(y_{n+1}) = z^{n+1}(1 - z^{-1})$, $\Delta^2 f(y_{n+1}) = z^{n+1}(1 - z^{-1})^2$, and so on. If we compare these differences with the corresponding ones for a predictor, we see an extra factor of z. This extra z propagates through the expressions, and finally shows up on the right side of the equation

that corresponds to equation 5 for predictor methods. It looks as follows:

$$\frac{\rho(z)}{\zeta^2} = \left[\frac{log(1-\zeta)}{\zeta}\right]^2 \left(\gamma_0^* + \gamma_1^*\zeta + \gamma_2^*\zeta^2 + \cdots + \gamma_k^*\zeta^k\right) \cdot z$$

If we expand this extra z as a series $(1-\zeta)^{-1} = 1 + \zeta + \zeta^2 + \ldots$, and combine it with the series $\left(\gamma_0^* + \gamma_1^*\zeta + \gamma_2^*\zeta^2 + \cdots + \gamma_k^*\zeta^k\right)$ we get the equation

$$\frac{\rho(z)}{\zeta^2} = \left[\frac{log(1-\zeta)}{\zeta}\right]^2 \left[\gamma_0^* + (\gamma_0^* + \gamma_1^*)\zeta + (\gamma_0^* + \gamma_1^* + \gamma_2^*)\zeta^2 + \ldots\right]$$

which is identical to the corresponding equation for the predictor, if we make the following identifications:

$$\begin{aligned} \gamma_0 &= \gamma_0^* \\ \gamma_1 &= \gamma_0^* + \gamma_1^* \\ \gamma_2 &= \gamma_0^* + \gamma_1^* + \gamma_2^* \\ &\vdots \end{aligned}$$

Now we can solve for γ_i^* in terms of γ_i and arrive at the desired result, formula 7.

Thus, the derivation of coefficients for multistep methods for 2nd degree equations has been completely automated. [12] The question now is how to choose the a_i parameters so as to get a good method for the problem we are interested in.

For long term integration of the solar system, we want methods that incur small error even when run at large stepsizes. Large stepsizes are necessary for otherwise the integration would take too long to complete. The accumulated truncation error is *proportional to the size* of the local truncation error which is given by

$$d_n \simeq C_k \cdot h^{k+1} \cdot f^{(k+1)}(\xi_n)$$

where C_k is the method's error constant, h the stepsize, and k the order of the method. C_k is given by

$$C_k = \frac{\gamma_k}{\gamma_0}$$

[12] A similar formulation can be easily made for multistep methods for 1st degree differential equations.

Hence, to minimize the accumulated error, we should try to use the highest order possible with the smallest error constant. This sounds very simple, but there are additional constraints. First, we want the method to be stable for the stepsize of interest. Secondly, the coefficients of the method should fit in the precision of the machine, typically 53 bits mantissa (IEEE64). We want the coefficients to be expressible as rational numbers with numerator and denominator smaller than 16 digits. The accurate approximation made by high order multistep methods is critically dependent on the exactness of the b_i coefficients. Finally, there are also other considerations such as having the $a_0, a_1, \ldots$ coefficients be nice numbers. For example, a few nice numbers are 2, -1, 3/2, and so on (numbers of small precision). Nice numbers simplify the arithmetic and reduce roundoff error.

In my search for optimal predictors I used the design criteria outlined above. But as I said previously, I came across a serious limitation between achievable order of accuracy and stability. I did not find any predictor of order 14 (or higher) which is stable on the Sun-Jupiter system, for stepsizes greater than 110 steps per orbit (40 days). The only exception to this was a method whose coefficients exceed 53 bits of precision. [13] The best predictor method, I have found, for the Sun-Jupiter problem, when the stepsize is 108 steps per orbit, is Stormer-13.

As far as the solar system is concerned, I haven't done any comparisons between different methods on the solar system. I have only made measurements of the error for Stormer-12 on the outer solar system, and it is similar to the error on

[13] One way to achieve 14th order of accuracy, under the given stability constraints, and using predictor-only methods, is to use two different predictors, for example Stormer-13 and S3N5-13 (described in a later section). On every integration step, both predictors are used to compute the new position — the computational cost is doubled. Let's call the two predicted positions y_1 and y_2, and the true position z. Then, we have $y_1 = z + C_1 A h^{14} + O(h^{15})$, and $y_2 = z + C_2 A h^{14} + O(h^{15})$, where C_1 and C_2 are known error constants, and A is a high derivative of the acceleration which is the same for both predictors. Using our knowledge of C_1 and C_2, we can eliminate the Ah^{14} terms and arrive at a new estimate for the true position which is accurate to order 15.

the Sun-Jupiter two-body system. This is not surprising because Jupiter is the most massive planet and has the smallest orbital period among the outer planets. I expect that Stormer-13 is one of the best choices for the outer solar system also.

I should add to the above that if we are interested in using a different stepsize than 108 steps per orbit, Stormer-13 may not be the optimal predictor. For larger stepsizes it becomes unstable on Jupiter's orbit, and for smaller stepsizes there are other predictors that are slightly more accurate than Stormer. I have found one such method and I have compared it experimentally against Stormer. It appears to be a very promising method for integrating the solar system. I will discuss it in detail in a later section. It belongs to a class of "three-point methods" which includes both more accurate and more stable methods than Stormer. The name "three point methods" comes from using a_0, a_1, a_2 as the parameters for this class of methods (see equation 2 for the meaning of these parameters).

STABILITY

It is appropriate now to say a few words about stability. There are basically two notions of stability for multistep methods, one for the case when the stepsize approaches zero, and the other for the case when the stepsize takes finite values. For practical purposes it's the second type of stability that matters, the first one being a mere prerequisite because we can't have the second one without the first. [14] To give a formal definition of stability, it is customary to refer to a particular test problem. In the case of second degree methods such as Stormer, the test problem is the harmonic oscillator. To say that a method is stable on a different problem, such as the two-body problem, one extends the definition of stability in an obvious way based on intuition. I'm not aware of a formal definition of stability for arbitrary problems that is universally agreed upon.

[14] The two types of stability have traditionally been misnamed strong and weak stability.

The definition of stability on the harmonic oscillator is as follows. When we replace a 2nd order differential equation with a 12th order difference equation (if we use Stormer-12), we introduce many extraneous solutions to our problem. These solutions are not part of the continuous problem, they are extraneous, and they should be kept small, much smaller than the true solution we are approximating. If the extraneous solutions grow large, the method becomes unstable.

The previous definition can be made more precise because we can find analytically both the true and the approximating solutions for the harmonic oscillator. Both the differential equation and the difference equation arising from a multistep method are linear. Their solutions are given by the roots of their characteristic polynomials. The differential equation has 2 conjugate roots, giving rise to a sinusoid. The difference equation has 13 roots (for Stormer-12). Two of these roots approximate the true sinusoidal solution (they are called principal roots), and they form a conjugate pair of unit magnitude. The rest of the roots give rise to extraneous solutions. If a method is stable, the extraneous roots must have magnitude less than 1, so that the extraneous solutions decay with time. [15] This is the definition of linear stability.

In a practical setting, one is interested in finding the largest stepsize for which a method remains stable on the harmonic oscillator. This can be done by trial and error as follows. We take the integration formula we are interested in and substitute $f(y) = -\omega^2 y$ to get the difference equation:

$$y_{n+1} = a_0 y_n + a_1 y_{n-1} + \cdots - (\omega h)^2 [b_0 y_n + b_1 y_{n-1} + \cdots + b_k y_{n-k}]$$

We then substitute specific values for the frequency ω and the stepsize h, and compute numerically the roots of the corresponding characteristic polynomial.

[15] Sometimes theoretical works claim that a method is stable even when there are extraneous roots on the unit circle, as long as these have multiplicity less than or equal to 2. But for practical purposes, this is misleading because the smallest increase in stepsize will push these extraneous roots outside the unit circle. An example of such a method appears on page 56.

If the roots lie inside or on the unit circle, the stepsize gives a stable formula for the chosen period of revolution; otherwise it doesn't. To find the maximum stable stepsize, we repeat this process for a range of values ωh (note that $\omega h = 2\pi \cdot$ *steps-per-cycle*), until we find the stepsize that switches from stability to instability or vice versa. It is also convenient (and it looks nice) to plot the computed roots against the unit circle. A couple of these plots appear at the end of the paper. I call them *Spider Plots* because of the shape in which the roots move outwards as h increases.

One more thing to mention is that computing the roots of polynomials arising from multistep methods poses no numerical problems. As far as I can tell from my experiments, a simple newton-root-finder does very well because the roots spread out nicely and symmetrically around the origin. When the stepsize is zero, some double roots occur, but they can be found analytically. For most methods there is a double root (the principal one) at $z = 1$, and there are more double roots at $z = 0$, all of which can be divided out of the polynomial.

Coming to the two-body problem now, in order to find the maximum stable stepsize on this problem, we can follow a trial and error approach just like the harmonic oscillator. However, there are a few new issues to worry about. For each trial stepsize, we need to perform a numerical integration to check the error. We also need a specific criterion to decide when a method is stable or unstable. And finally we need to consider whether the eccentricity and the orbital period of the two-body problem we are integrating, affect in any way the maximum stable stepsize.

First, I will answer the question of defining stability on the two-body problem. I have considered a method to be stable on elliptic two-body systems (eccentricity less than 1) if the error does not grow in an explosive manner, if the computed trajectory keeps going around the true ellipse more or less, even if it lags behind in

angle by a whole revolution. This seems to correspond rather well to the intuitive notion of stability, and it's easy to check for experimentally. When a method becomes unstable, it very quickly breaks away from the true elliptical solution, the position error becomes many times larger than twice the semimajor axis.

The stepsizes to try when looking for the stability boundary on a two-body system should be close to the stepsize for which the method becomes unstable on a harmonic oscillator of the same orbital period. Typically, the stability boundary on a two-body system is smaller than the one on the harmonic oscillator; how much smaller depends on the eccentricity. For example, the stability boundary for Stormer-13 on a two-dimensional harmonic oscillator of orbital period equal to 4334 days is about 45 days. The corresponding boundary on a two-body problem of eccentricity .05 (Jupiter) and same orbital period is approximately 40 days. A surprising fact is that the higher the eccentricity is, the heigher the stability boundary is (up to an eccentricity of .6). I found this by testing Stormer-13 on two-body systems of orbital period equal to 4334 days and different values of eccentricity. The following table shows the maximum stable stepsizes (accurate to within a day) as a function of eccentricity.

Eccentricity	0	.05	.1	.2	.3	.4	.5	.6	.7
Stepsize (days)	39	40	40	40	41	42	43	45	32

In all of these tests I checked the error for about 200 hundred revolutions and declared instability when the error became huge with respect to the semimajor axis. For a semimajor axis equal to 5 units (the Sun-Jupiter semimajor axis is 5 A.U.), huge means about 1000 units. But it actually suffices that the error (in position) exceeds by some amount twice the semimajor axis. The error of twice the semimajor axis is the error that occurs when the computed solution is at perihelion and the true solution is at aphelion (or vice versa). Anything larger than this value is an indication that the computed trajectory has broken away

from the true ellipse. Once this happens, it usually takes zero time before the error shoots up in the range of 1000 or 2000 units.

With regard to the question of whether the orbital period affects stability, the answer is no. Although the maximum stable stepsize doubles when we double the orbital period, the number of steps per cycle is the same in both cases. In this sense, stability is not a function of orbital period, but a function of the number of steps per cycle. I have tested this experimentally by running Stormer-13 on two-body systems of eccentricity .05 and .3 . The maximum number of steps per cycle for which Stormer-13 is stable, remains constant if we scale the period and the stepsize by the same amount.

To finish the discussion on stability, I will mention one more thing. It's a most basic question, "Why does one care about stability in the first place? What is the importance of knowing the maximum stable stepsize?" The answer may seem obvious: to make sure that a method does not compute incorrect answers. But there is something more to be said about it. First, I think that the maximum stable stepsize is *not important if we are integrating stiff two-body systems*, that is systems of large eccentricity (larger than .4). This is because the error in integrating such systems with uniform stepsize multistep methods, is so large per orbit, that one never uses a stepsize close to the stability boundary. For stiff problems, one uses much smaller stepsizes, where questions of stability do not arise.

The case where stability becomes a critical factor is when we are integrating smooth two-body systems, such as the Sun-Jupiter system. For such systems, the truncation error is very small, even for large stepsizes. If the order of a method is high enough (say 13), then the truncation error competes with roundoff even for stepsizes close to the stability boundary. Large stepsizes are desirable because they reduce the roundoff error and the total running time. For this reason, the stepsizes close to the stability boundary are the most popular ones to use in these

cases. Therefore, it is important to know the maximum stable stepsize, and it is also important to use an integration method which allows for the largest stable stepsize, without compromising other properties.

6 THE CLASS OF 3-POINT METHODS

In my search for new multistep integrators I have found a class of three-point methods which has nice and well-understood properties, and may be useful in practice. This class of methods includes both Stormer and the more accurate method S3N5 that I've mentioned occasionally in previous parts of this paper. The general form of a predictor method in this class looks as follows:

$$y_{n+1} = a_0 y_n + a_1 y_{n-1} + a_2 y_{n-2} + h^2 \left[b_0 f(y_n) + b_1 f(y_{n-1}) + \cdots + b_k f(y_{n-k}) \right]$$

The three coefficients a_0, a_1, a_2 must satisfy $y(t) = 1$ and $y(t) = t$; hence there is one free parameter to determine the method, and it's convenient to choose a_2 as the free parameter. The following relations hold:

$$a_0 = 2 + a_2 \qquad \text{and} \qquad a_1 = -(1 + 2a_2)$$

Setting $a_2 = 0$ gives Stormer and setting $a_2 = -1/2$ gives S3N5 which looks as follows:

$$y_{n+1} = \frac{1}{2}(3y_n - y_{n-2}) + h^2 \left[b_0 f(y_n) + b_1 f(y_{n-1}) + \cdots + b_k f(y_{n-k}) \right] \qquad (8)$$

Like Stormer, it uses only two previous points (a_1 cancels out), which means fewer arithmetic operations. The coefficients of S3N5 are listed in the appendix, page 78.

To compare the accuracy of Stormer and S3N5, we can look at their respective error constants, which are given in Table 2. For example, to compare Stormer and S3N5 order 10, we read under column 10. The error constants are 5.9e-2 and 3.9e-2 respectively, and their ratio is about 2/3. Hence, we expect S3N5 to be about 2/3 more accurate than Stormer, as far as truncation error is concerned. I have verified this experimentally.

Figure 10 on page 90 shows the position and energy errors for Stormer and S3N5 order 10 on the Sun-Jupiter system. After 4096 revolutions the position errors are respectively 9E-6 and 6E-6 A.U. (Astronomical Units) [16], which is exactly in the

[16] See page 10 for an explanation of units.

	7	8	9	10	11	12	13	14
Stormer	6.5e-2	6.3e-2	6.1e-2	5.9e-2	5.8e-2	5.6e-2	5.5e-2	5.4e-2
S3N5	4.3e-2	4.1e-2	.04	3.9e-2	3.8e-2	3.7e-2	3.6e-2	3.5e-2
S35	.13	.13	.12	.12	.12	.11	.11	.11
H615	1.5e-2	1.5e-2	1.4e-2	1.4e-2	1.4e-2	1.3e-2	1.3e-2	1.3e-2
Cowell	-2.7e-3	-2.4e-3	-2.1e-3	-1.8e-3	-1.6e-3	-1.5e-3	-1.3e-3	-1.2e-3
H621	-4.8e-4	-4.3e-4	-3.9e-4	-3.5e-4	-3.2e-4	-2.9e-4	-2.7e-4	-2.5e-4

Table 2: The Error Constants of some multistep methods, for orders of accuracy between 7 and 14.

ratio 2/3. The nice agreement is due to the fact that the error is dominated by truncation.

The next thing to consider is stability. Before looking at a complete linear analysis as outlined in the previous section, it is worth noting a nice property of all methods in the class a_0, a_1, a_2. For zero stepsize (strong stability) the roots of the error equation are a double root at 1, a single real root at a_2 (this is why I chose a_2 as a parameter for these methods), and 10 more roots, for a 12th order method, at zero. The location of the real root at a_2 characterizes completely these methods. From my experiments, I have found that as a_2 varies from 0 towards -1, we get methods that are increasingly more accurate than Stormer and less stable. At $a_2 = -1$, we have extreme accuracy, but the region of stability is practically null, as soon as the stepsize becomes non-zero the real negative root at -1 jumps outside the unit circle.

As a_2 varies from 0 towards 1, we get methods that are increasingly less accurate and more stable than Stormer. Setting $a_2 = 1/2$ gives a method which is listed as S35 in the table of error constants. At order 13 its error constant is .11,

which is not very attractive. Yet, it is interesting to consider the following: S35-14 is stable on the harmonic oscillator at 135 steps per cycle, but Stormer-14 is unstable at that stepsize. To use Stormer, one must go down one order of accuracy to Stormer-13, which means approximately an increase in error by 2π over the number of steps per cycle. The increase in error is about 1/20. This is because the accumulated truncation error is proportional to $(\omega h)^k$ for a kth order method. Hence, though Stormer-13 has an error constant of 5.5e-2, S35-14 has, effectively, an accuracy of .11/20 or 5.5e-3 which is much better.

The preceding scenario comparing Stormer and S35 is, of course, unrealistic. In practice there is some freedom in choosing the stepsize. Further, if the computer uses IEEE64 double precision arithmetic, then neither the coefficients of Stormer-14 nor the coefficients of S35-14 are representable as rational numbers on a common denominator. For S35-14 we need 3 more bits of mantissa. The common denominator form is necessary for high order methods. I am also assuming that the coefficients have been computed exactly using rational arithmetic. The accuracy of high order methods depends critically on the exactness of the b_i coefficients.

The common denominator is used as follows: We first multiply the accelerations with the numerators of the $b_0, b_1, \ldots$ coefficients, sum up the products, and then scale by h^2/d where d is the common denominator. This has two advantages over the alternative formulation of using divided-out (numerator over denominator) coefficients. First, it preserves as much as possible the precision of the coefficients, which is vital to achieving high order accuracy. Secondly, it is in general better to factor out any precision possible when computing sums in finite arithmetic. The factored-out sum is always closest to the infinite precision sum. Indeed, a few experiments that I have done on this, show great difference in error between the two formulations, the divided-out form behaves at least like an order of accuracy

lower than the factored-out form.

Coming back to stability analysis, it is interesting to observe what happens to the roots of the error equation as the stepsize increases. Figure 11 on page 91 shows the location of the roots against the unit circle for Stormer-12 and S3N5-12. Figure 12 shows the corresponding plot for Stormer-13. There are 9 sets of 13 roots on each picture corresponding to 9 values of the stepsize. When the period of oscillation is 4334 days, the stepsize values are .00001, .001, .1, 2, 10, 30, 40, 60, and 90 days. Alternatively they can be expressed as 4e8, 4e6, 43344, 2167, 433, 144, 108, 72, and 48 steps per cycle.

As the stepsize increases, the double root at $z = 1$ splits into 2 complex conjugate roots, which move along the unit circle. These are the principal roots, which approximate the true solution. The roots at the origin, all of which give extraneous solutions, initially grow outwards in almost co-centric circles. For larger stepsizes they distort into a shape that looks like a spider. This is true, more or less, for all 2nd-degree multistep methods I have looked at. For this reason, I named the plots of the roots against the unit circle, *Spider plots.*

EXPERIMENTAL COMPARISON: S3N5 VERSUS STORMER

To finish the comparison between S3N5 and Stormer, I will describe one more experiment I have done: a comparison of the error when each method is applied to the Sun-Jupiter system. In this test, I run both methods at 12th order of accuracy, and a stepsize of 32 days. I did the comparison first using double precision arithmetic, and secondly using a scheme in which some arithmetic operations are done in quad precision. This technique, which reduces the roundoff error, is described in Section 7. In both tests, S3N5 turned out to be more accurate than Stormer. The tests are interesting not only for comparing the two methods, but also because they show something about the peculiar character of roundoff.

Figures 13 and 14 on page 13 show the error for the double precision test. The strange looking cancellation in the position error that occurs for both methods is not accidental. It is due to a peculiar interaction between roundoff and truncation error, that occurs at the initial stages of the integration, and disappears as truncation grows and dominates the error. What seems to be happening, is that the computed solution starts spiraling inwards (the energy error is negative), which causes it to move faster than the true solution. But soon thereafter, it changes its mind and spirals outwards (the energy error crosses zero and becomes positive), slowing down and catching up with the true solution. In the long run, the computed solution for 12th order predictors always (in all of my tests) ends up spiraling outwards of the true orbit.

I have seen this particular cancellation only for 12th order of accuracy predictors, using double precision arithmetic, and only for certain ranges of eccentricity. I have observed it for eccentricities .05 and also .24, including nearby values. As the eccentricity increases or decreases away from .05 or .24, the cancellation becomes smaller and smaller until it is no longer visible. Another interesting thing is that the derivative of the position error during the cancellation bump at eccentricity .05 (figure 13), is approximately linear. The error curve is approximately a t^2 turned upside down. For eccentricity .24 (larger local truncation error), this error curve is a perfect t^2. Such an example can be seen in figure 15 on page 95, which shows the position error, the derivative of the position error (computed by subtracting successive samples), and the energy error. Incidentally, figure 3 on page 83 shows a "local view" of the error for the same integration. It shows the error during 2 revolutions which fall inside the cancellation bump.

In my comparison of Stormer and S3N5 using quad precision, no cancellation bump occurs. Figures 16, 17, and 18 starting on page 16 show the error in 3 stages of the integration corresponding to 16x1024, 64x1024, and 200x1024 revolutions. The first thing to note is that despite the quad precision arithmetic, the roundoff

error is still quite dominant initially, at least for S3N5. This is because the truncation error of S3N5 is smaller than Stormer's. The wandering up and down of S3N5's energy error is an indication of roundoff. The interesting thing in these figures is how S3N5 starts doing better (smaller position error) than Stormer, then in the second figure it does worse, and finally in the third figure it comes back to win.

What happens is that toward the end of the first figure (around 16x1024 revolutions), S3N5 incurs a large error in semimajor axis (probably an effect of roundoff). It can be seen by the sudden increase in energy error, which makes S3N5 energy error be larger than Stormer's. This apparently random event develops in the second figure, and makes S3N5 lag behind Stormer for a while. However, by this point in time, both methods have clearly entered the t^2 truncation error growth, which makes the error predictable.

According to my theory of truncation error growth, S3N5's error must grow like At^2, and Stormer's error must grow like Bt^2, with $A/B = 2/3$ being their respective error constants. A quick calculation shows that given the known errors at time 64x1024 revolutions, the position error curves for the two methods should cross each other at approximately 200x1024 revolutions. In fact, I made exactly such a calculation when I was doing this experiment because it takes several days to run an integration, and I needed to estimate how long to run. Indeed, in the third figure, the crossover occurs a little before 200x1024 revolutions. From that time on, the errors approach the ratio 2/3, while the absolute difference of the errors grows at a t^2 rate.

THE NEGATIVE ROOT GOES UNSTABLE

Having finished the comparison between Stormer and S3N5, one more fact about spider plots should now be mentioned. In all the spider plots I have seen, the instability occurs by the real negative root growing larger than -1. This includes

predictor form looks like

$$y_{n+1} = 2y_{n-1} - y_{n-3} + h^2 \left[b_0 f(y_n) + b_1 f(y_{n-1}) + \cdots + b_k f(y_{n-k}) \right]$$

and there is of course a corresponding corrector. I'll refer to them as H615 and H621. In the table of error constants presented previously, one can see that they both have impressive error constants, 1e-2 for the predictor and 3e-4 for the corrector. It is further interesting to note that the coefficients for the predictor fit in IEEE64 up to order 17. H615 would be an impressive method, if it wasn't for the fact that it's stable only for zero stepsize.

The strong roots, that is zero stepsize, are a double root at 1, and a double root at -1. The double root at -1 immediately splits into two real roots for the slightest nonzero stepsize, one root moving away from the unit circle with the other moving towards zero. Figure 19 on page 99, shows two spider plots of which the bottom one corresponds to H615 order 4. The figure shows that the instability occurs even at order 4, no more roots are necessary to push the negative root outside the unit circle.

Incidentally, the top spider plot in figure 19 corresponds to Cowell-13. I included it because it may of interest to some readers. It shows that correctors are much more stable than predictors, the spider signature is shrunk and restricted close to the origin. If one can afford the cost of recomputing the accelerations, a corrector has a lot of advantages over a predictor-only scheme. As a matter of fact, Cowell remains stable up to order 18 on the harmonic oscillator for stepsizes around 135 steps per cycle. However, its coefficients fit in IEEE64 (53 bits mantissa) only up to order 15.

---

the use of the unstable predictor with a Cowell corrector, a combination which may be stable if iterated enough times. I haven't tested it. There is also a mistake in Hamming's book concerning this method. It says "The characteristic roots of the predictor are $1, -1, i, -i$ and give no trouble." This is clearly false.

predictor form looks like

$$y_{n+1} = 2y_{n-1} - y_{n-3} + h^2 \left[b_0 f(y_n) + b_1 f(y_{n-1}) + \cdots + b_k f(y_{n-k}) \right]$$

and there is of course a corresponding corrector. I'll refer to them as H615 and H621. In the table of error constants presented previously, one can see that they both have impressive error constants, 1e-2 for the predictor and 3e-4 for the corrector. It is further interesting to note that the coefficients for the predictor fit in IEEE64 up to order 17. H615 would be an impressive method, if it wasn't for the fact that it's stable only for zero stepsize.

The strong roots, that is zero stepsize, are a double root at 1, and a double root at -1. The double root at -1 immediately splits into two real roots for the slightest nonzero stepsize, one root moving away from the unit circle with the other moving towards zero. Figure 19 on page 99, shows two spider plots of which the bottom one corresponds to H615 order 4. The figure shows that the instability occurs even at order 4, no more roots are necessary to push the negative root outside the unit circle.

Incidentally, the top spider plot in figure 19 corresponds to Cowell-13. I included it because it may of interest to some readers. It shows that correctors are much more stable than predictors, the spider signature is shrunk and restricted close to the origin. If one can afford the cost of recomputing the accelerations, a corrector has a lot of advantages over a predictor-only scheme. As a matter of fact, Cowell remains stable up to order 18 on the harmonic oscillator for stepsizes around 135 steps per cycle. However, its coefficients fit in IEEE64 (53 bits mantissa) only up to order 15.

---

the use of the unstable predictor with a Cowell corrector, a combination which may be stable if iterated enough times. I haven't tested it. There is also a mistake in Hamming's book concerning this method. It says "The characteristic roots of the predictor are $1, -1, i, -i$ and give no trouble." This is clearly false.

To finish this section, one last thing about predictor-corrector schemes is in order. A predictor-corrector method typically has the form PE(CE), meaning that the new position is predicted, the new acceleration is evaluated and corrected, and perhaps the EC step is repeated over and over again to achieve greater accuracy. On the other hand, a predictor-only method consists of a single PE step. An interesting question is why we do not adopt a PEC strategy, which seems to have the advantages of a corrector, but evaluates the acceleration only once (force calculation is the most costly step in integrations of the solar system). The idea is too good to be true. I tried it on the two-body problem, and I ended up with an unstable method. The reason is, probably, that avoiding an extra E step at the end of every step, makes the positions and accelerations slightly inconsistent. The feedback in the method quickly sees the inconsistency, and accentuates it into an instability. I should add that in an effort to avoid the instability, I tried to substitute the E step with a linear interpolation for the new acceleration (this is zero cost compared to exact force calculations). However, neither this idea nor a few others that I tried, worked.

7 WAYS OF REDUCING ROUNDOFF

Roundoff error belongs to those parts of numerical analysis that are least well understood. Roundoff is generally considered to be something undesirable, something that should be eliminated as much as possible; hence the justification for this section. Yet, I know of many example integrations where reducing the roundoff error by employing the techniques of this section, results in increased accumulated error. I presented some of these integrations in the previous section, in comparing Stormer and the new method S3N5 using double and quad precision arithmetic. Perhaps, the careful reader noticed that the double precision integrations resulted in smaller accumulated error than the quad precision integrations. The reason for this effect is, probably, the cancellation of angular error caused by first spiraling in and then out of the true orbit, which occurred only in the double precision integrations.

Incidentally, the difference in accumulated error between double and quad integrations in these examples (Stormer and S3N5) does not diminish as time goes on. The reason is that after the initial transient period where roundoff plays a dominant role, the truncation error develops and dominates the error. From then on, quad or double precision makes no difference, the error is the same. So in a way, the initial cancellation of angular error in the double precision version can be viewed as giving the double version a head start over its quad counterpart. It's as if the double precision integration started a little later in time than the quad precision integration. Mathematically, this means the following:

$$\begin{aligned} \text{quad-error} &= A \cdot t^2 \\ \text{double-error} &= A \cdot (t - C)^2 \\ \text{ratio} &\rightarrow 1 \\ \text{difference} &= (2AC)t + AC^2 \end{aligned}$$

I have verified both of these formulas experimentally. For example, the error ratio

of double and quad Stormer-12 on the Sun-Jupiter system (32 days stepsize) after 200x1024 revolutions is about 1.15 and diminishes slowly as time goes on. On the other hand, the error difference grows *linearly* with time.

In contrast to this, and to gain confidence in the preceding formulas, I have also measured the growth of the error difference between S3N5 and Stormer. Naturally enough, the error difference grows *quadratically* with time in this case, and the error ratio approaches the ratio of the error constants, 2/3. In the comparison examples presented in the previous section, the error ratio at 10th order of accuracy is 2/3 (= .67), right from the beginning of the integration, while the ratio at 12th order of accuracy is .73 after 200x1024 revolutions. The deviation of the ratio at order 12 is probably due to the peculiar angular cancellation, which occurs at slightly different points in time for the two methods (see figure 13).

Coming back to the original discussion regarding whether roundoff is desirable or not, there are many integration examples where reducing the roundoff error, reduces the total accumulated error. For example, all integrations I have seen, using multistep methods of order 13, are like that. In these examples, there is a considerable length of time at the start of the integration, during which roundoff is by far the dominant source of error. I must also add that when the truncation error grows eventually large enough, no peculiar error cancellation occurs as in 12th order methods, as far as I have seen. It's conceivable, though, that there are cases where cancellations occur even for order 13 methods; possibly for large eccentricity where the truncation error is as large as in 12th order methods.

In any way, it is clear that there are cases where reducing the roundoff error is advantageous, just as there are some cases where an increased level of roundoff is desirable. In the rest of this section, I describe two techniques for reducing roundoff: quad precision arithmetic and the summed form transformation.

Roundoff exists because computers have a finite word-length, and hence a fi-

nite precision for representing numbers. To reduce roundoff, an obvious thing to do is to increase the precision. Since modern computers use double precision arithmetic, it is natural to increase the precision by simulating quad precision in software. The algorithms for reducing quad arithmetic operations into double precision operations are not very difficult to invent or to understand. A good reference paper, which describes clearly the algorithms, is T.Dekker: "A floating-point technique for extending the available precision," *Numer. Math. 18*, 224-242, 1971. For shortness of breath, I have not included them in this paper. However, I have some practical advice to give to the reader about programming these algorithms. It comes from painful experience.

There are certain pitfalls in implementing the quad algorithms using the C programming language, on a machine with a floating-point co-processor (e.g. MC68881). The quad algorithms work only if the intermediate computations are *rounded correctly* to double precision. Most floating-point chips (such as the MC68881) store the numbers internally in extended precision, and the C compiler will, on occasion, try to optimize the code and avoid re-loading a rounded result from memory, when it sees that the result resides in a floating point register from a previous operation. If this happens (for example it will happen in the normalization step of adding two numbers into a quad precision sum), then quad precision fails badly.

The scenario just described should be classified as a C compiler bug because it violates the definition of the C programming language. Worse, yet, is the fact that this bug does not manifest itself in a small program for testing the quad algorithms; the C compiler does not try to optimize such cases. It only optimizes code that gets executed many times, such as inside tight loops of the integration code. The unwary user will be surprised by the results. I was. It took me quite a while before I looked at the assembly language code to discover what was actually happening. To make things work right, I inserted a dummy operation that cleared

the floating-point register and forced a load of the rounded intermediate result from main memory.

Coming back to the question of using quad precision for numerical integration, there are a few things to consider. Using quad arithmetic for all operations in an integration step is very costly. In particular, quad multiplication is forbiddingly expensive. Fortunately, we can have most of the advantages of quad precision without doing any multiplications or a lot of quad additions, which are required in the acceleration part of 2nd degree multistep methods. The idea is to notice that the acceleration sum, which is added to a linear combination of the position values, is smaller in size than the positions, for most of the time (I will justify this shortly). Therefore, the roundoff error in the positions is much greater than the roundoff error in the accelerations. Hence, it is cost-effective to use quad precision for the positions, and double precision for the accelerations.

For example, to use the preceding scheme with Stormer, we store the position values y_n, y_{n-1} in quad precision, and we perform one quad subtraction $(2y_n - y_{n-1})$, and one quad-plus-double addition, when we add to this result the acceleration sum. The acceleration sum is computed in double precision. In my quad precision integrations, mentioned in this paper, I have used exactly this scheme.

Now, I must justify the statement that "the acceleration sum is smaller in size than the positions, for most of the time." The acceleration sum is the expression $h^2\,[b_0 f_n + \ldots + b_k f_{n-k}]$. My statement says that this expression is smaller in size than the linear combination of the positions in the integration formula. For Stormer, the linear combination of the positions is $(2y_n - y_{n-1})$. The idea is to view these two expressions as "the old position plus the increment". $(2y_n - y_{n-1})$ has the size of the old position, and $(h^2 \cdot \textit{acceleration})$ is the increment. Viewing the problem this way, it becomes a more general statement. It applies to 1st degree methods, also, when "acceleration sum" is replaced by "velocity sum". Further,

the problem becomes *unitless*. We do not have to worry about specific units for position and time.

To see why the increment is smaller than the old position, for most of the time, it's convenient to sketch (or imagine) a sine wave versus time. That's what the solution of smooth orbital problems looks like, approximately. The next step is to mark about 120 time intervals (or fewer if the sketch does not allow so much detail), all inside one period. This makes the proof obvious: The height of the curve at every mark is "the old position", and the difference between marks is "the increment". Clearly, the only time when the increment is comparable to the old position is at zero crossings. During the rest of the orbit, the old position is much larger than the increment. Hence, the roundoff in the old position is much larger than the roundoff in the increment. This finishes the proof. Incidentally, I chose 120 intervals because it's a typical stepsize for Sun-Jupiter integrations, it corresponds to 36 days.

THE SUMMED FORM OF MULTISTEP METHODS

The second technique for reducing roundoff error is the summed form. It is an algebraic transformation of the multistep formula that results into a new formula, which has fewer arithmetic operations in the position part of the formula (the $a_0, a_1, \ldots$), and more arithmetic operations in the acceleration part of the formula (the $b_0, b_1, \ldots$). The transformation is desirable because roundoff in position values is more significant than roundoff in the acceleration sum, as I explained when discussing quad precision integrations.

For Stormer, the original and the summed form equations are as follows:

$$\begin{aligned} y_{n+1} &= 2y_n - y_{n-1} + h^2\left[b_0 f_n + b_1 f_{n-1} + \cdots + b_k f_{n-k}\right] \\ y_{n+1} &= y_n + h^2\left[b_0 F_n + b_1 F_{n-1} + \cdots + b_k F_{n-k}\right] \end{aligned} \tag{9}$$

where the F_i are the *summed accelerations*, defined by $F_i = F_{i-1} + f_i$, and the $f_i = f(y_i)$ are the actual accelerations. One can see that the subtraction $2y_n - y_{n-1}$ of the original Stormer formula is not present in the summed form. This subtraction is a major source of roundoff. An extra addition is needed at every integration step to compute the summed accelerations, but this addition does not cause any numerical problems.

Another thing about the summed form is that it requires a special computation at the beginning, in order to start. The first summed acceleration is given by $F_0 = F_{-1} + f_0$ where F_{-1} is defined so as to make the summed form consistent. The extra computation is needed for computing F_{-1}. For Stormer 12th order, F_{-1} is given by the following formula:

$$F_{-1} = \frac{1}{h^2}\left[y_{12} - y_{11} - h^2(b_0 g_{11} + b_1 g_{10} + \cdots + b_{11} g_0)\right] \tag{10}$$

$$
\begin{aligned}
& & g_{11} &= f_{11} + g_{10} \\
& & g_{10} &= f_{10} + g_9 \\
& \text{where} & &\vdots \\
& & g_1 &= f_1 + g_0 \\
& & g_0 &= f_0
\end{aligned}
$$

The b_i coefficients in the preceding formula are the same b_i coefficients as in formula 9.

I will now show how to derive the preceding formulas. By following the same steps, one can derive the summed form for any of the methods discussed in this paper. The first key idea is to write down the original Stormer formula for 3 or 4 successive points, as in the following example given for Stormer-12, and then to sum up the equations. The more a_i coefficients a method has, the more equations we need to sum up. Stormer has 2 coefficients, a_0 and a_1, so 3 equations are enough to show the pattern.

$$
\begin{aligned}
y_{13} - 2y_{12} + y_{11} &= h^2(\ldots) \\
y_{14} - 2y_{13} + y_{12} &= h^2(\ldots) \\
y_{15} - 2y_{14} + y_{13} &= h^2(\ldots) \\
\hline
(y_{15} - y_{14}) + (y_{12} - y_{11}) &= h^2(\ldots)
\end{aligned}
$$

Therefore, the summed form equation looks as follows:

$$(y_{n+1} - y_n) + (y_{12} - y_{11}) = h^2(\ldots)$$

The $(y_{12} - y_{11})$ term is a constant which gets absorbed in the right-hand side of the equation by introducing an appropriate F_{-1}, which also "fills in some initial summed accelerations" to give the final form:

$$y_{n+1} = y_n + h^2 \left[b_0 F_n + b_1 F_{n-1} + \cdots + b_k F_{n-k}\right]$$

where $F_n = f_n + F_{n-1}$. I am omitting to write down explicitly the steps in summing up the accelerations (the right-hand side) because the result is identical for all

methods discussed in this paper. It's more important to verify that the given right-hand side works. For this, we expand the summed accelerations $F_n = f_n + F_{n-1}$, separate the terms into two groups, and notice that the second group (as shown below) is the summed formula for the preceding $(n-1)$ index:

$$\begin{aligned}
y_{n+1} - y_n &= h^2 [b_0 F_n + b_1 F_{n-1} + \cdots + b_k F_{n-k}] \\
y_{n+1} - y_n &= h^2 [b_0 f_n + b_1 f_{n-1} + \cdots + b_k f_{n-k}] + \\
&\qquad h^2 [b_0 F_{n-1} + b_1 F_{n-2} + \cdots + b_k F_{n-k-1}] \\
y_{n+1} - y_n &= h^2 [b_0 f_n + b_1 f_{n-1} + \cdots + b_k f_{n-k}] + (y_n - y_{n-1}) \\
\hline
y_{n+1} &= 2y_n - y_{n-1} + h^2 [b_0 f_n + b_1 f_{n-1} + \cdots + b_k f_{n-k}]
\end{aligned}$$

In other words, if the summed formula is correct (equivalent to the original formula) for index $(n-1)$, then it's also correct for index (n). This is the inductive step of the proof. The basis step is to show that the summed formula is correct for index $(k-1)$, where k is the order of the method. Notice that when the order of the method is k, the initial positions to start the method are $y_0, y_1, \ldots, y_k$. For Stormer-12, we wish to show that the following equation is true:

$$y_{12} = y_{11} + h^2 [b_0 F_{11} + b_1 F_{10} + \cdots + b_{11} F_0 + b_{12} F_{-1}] \qquad (11)$$

This we do by definition, by choosing appropriately the F_{-1} term. If we expand the summed accelerations $F_{11}, F_{10}, \ldots, F_0$ in the preceding equation, successively until there are no summed accelerations left, then we are left with an algebraic equation in one unknown, the F_{-1} term. Solving for F_{-1} gives formula 10, and also finishes the proof that the summed form is correct.

Solving for F_{-1} is the second key step in deriving the summed form; the first one was to write down the equations and sum them up. Now, to solve for F_{-1}, it is convenient to do so for orders, say 2 and 3, and see the pattern emerging. This saves some effort over the alternative approach of solving equation 11 (order 12)

directly. For order 2, we have the following:

$$k = 2 \quad y_2, y_1, y_0 \text{ are given.}$$

$$y_2 = y_1 + h^2 \left(b_0 \underbrace{F_1}_{f_1+f_0+F_{-1}} + b_1 \underbrace{F_0}_{f_0+F_{-1}} + b_2 F_{-1} \right)$$

$$y_2 = y_1 + h^2 \left(\begin{array}{l} (b_0+b_1+b_2)F_{-1} + \\ b_0 f_1 + b_1 f_0 + \\ b_0 f_0 \end{array} \right)$$

hence

$$F_{-1} = \tfrac{1}{h^2}\left(y_2 - y_1 - h^2(b_0 g_1 + b_1 g_0)\right)$$

where
$$\begin{aligned} g_1 &= f_1 + g_0 \\ g_0 &= f_0 \end{aligned}$$

Similarly for order 3:

$$k = 3 \quad y_3, y_2, y_1, y_0 \text{ are given.}$$

$$y_3 = y_2 + h^2 \left(b_0 \underbrace{F_2}_{f_2+f_1+f_0+F_{-1}} + b_1 \underbrace{F_1}_{f_1+f_0+F_{-1}} + b_2 \underbrace{F_0}_{f_0+F_{-1}} + b_3 F_{-1} \right)$$

$$y_3 = y_2 + h^2 \left(\begin{array}{l} (b_0+b_1+b_2+b_3)F_{-1} + \\ b_0 f_2 + b_1 f_1 + b_2 f_0 + \\ b_0 f_1 + b_1 f_0 + \\ b_0 f_0 \end{array} \right)$$

hence

$$F_{-1} = \tfrac{1}{h^2}\left(y_2 - y_1 - h^2(b_0 g_2 + b_1 g_1 + b_2 g_0)\right)$$

where
$$\begin{aligned} g_2 &= f_2 + g_1 \\ g_1 &= f_1 + g_0 \\ g_0 &= f_0 \end{aligned}$$

In both of the above derivations, I have used the fact that for any order of accuracy k, the following identity holds $(b_0 + b_1 + \ldots + b_k) = 1$. This can be shown by applying the original Stormer formula to the polynomial $y(t) = t^2$ which must be integrated exactly (setting $h = 1$ is convenient).

With regard to the initial goal of proving formula 10, which gives F_{-1} for order 12, we are now done. The pattern shown in the two previous derivations makes it obvious. This finishes the derivation of the summed form for Stormer. The same approach can be used to derive the summed form for any method discussed in this paper. In the case of correctors, the only change is the accelerations (both the f_i, the F_i, and the g_i) start at index $(n+1)$ instead of index (n).

MORE COMMENTS ABOUT THE SUMMED FORM

A slightly more general derivation of the summed form than the one I have given, can be found in Henrici's book (1962) chapter 6. Also, Henrici uses an extra parameter in the summed form, the so called α factor, which can be chosen freely to give better numerical performance. By ignoring this extra factor in my discussion, I have effectively chosen it to equal h, the stepsize. This avoids one extra arithmetic operation in Henrici's formula $F_i = F_{i-1} + \frac{h}{\alpha} f_i$, and it also factors out of the acceleration sum as much precision as possible. I have discussed previously why factoring out precision is desirable. I should add that there is an error in Henrici's book, page 330, formula (6-147a), which gives the term F_{-1} (Henrici calls this term H) for Stormer. The correct formula in his notation (using differences) should be

$$\alpha h H = \nabla y_{k-1} - h^2(\sigma_0 f_{k-2} + \sigma_1 f_{k-2} + \sigma_2 \nabla f_{k-2} + \cdots + \sigma_{k-1} \nabla^{k-2} f_{k-2})$$

As far as experimental results on the summed form are concerned, I have tested it for Stormer-12 and Stormer-13. As expected, the summed form reduces the roundoff error. For example, the error cancellation shown in figure 13 on page 13, for the original Stormer-12 on the Sun-Jupiter system, does not occur for the summed form Stormer-12, just like it doesn't occur for the quad Stormer-12. The accumulated error for the summed form on this example is 1.4e-6 A.U. as opposed to .73e-6 A.U. for the original Stormer-12 (double precision). In contrast

to this, the summed form for Stormer-13 does better than the original Stormer-13 (double precision), just like the quad version Stormer-13 does better than the original Stormer-13 (double version).

With regard to whether the summed form Stormer-13 is better than quad Stormer-13, I have no clear answer. I have seen examples where each scheme does better than the other. For example, on the Sun-Jupiter system the summed form has smaller position error than the quad scheme, when the stepsize is 32 days. Yet, quad does better when the stepsize is 34 days. The differences are due to interactions between roundoff and truncation, which I do not understand. One thing is clear: Both the summed form and the quad scheme reduce roundoff. Also, in practical situations, one should not forget the fact that the summed form is much less costly in execution time than the quad scheme.

To finish the discussion on the summed form, I will make one more comment. One can apply the summed form transformation more than once, that is one can "sum up" again the summed form equation. There is nothing special about applying the transformation on the original multistep formula, it can be applied again, with similar effects: to exchange one operation involving position values with one operation involving accelerations. However, I am not aware of any practical application of this.

Applying the summed form on Stormer twice, gives an equation of the form $y_{n+1} = h^2(\ldots)$, which is far from being desirable. The acceleration sum, which involves lots of arithmetic operations, becomes the dominant source of roundoff error. Perhaps, applying the summed form twice or more on methods involving many a_i coefficients, might be useful.

For completeness, I have listed below the summed form for the new method

S3N5 (see page 50). The original and the summed form equations are as follows:

$$\begin{aligned} y_{n+1} &= \tfrac{1}{2}(3y_n - y_{n-2}) + h^2\left[b_0 f_n + b_1 f_{n-1} + \cdots + b_k f_{n-k}\right] \\ y_{n+1} &= \tfrac{1}{2}(y_n + y_{n-1}) + h^2\left[b_0 F_n + b_1 F_{n-1} + \cdots + b_k F_{n-k}\right] \end{aligned} \tag{12}$$

where the summed accelerations are defined by $F_i = F_{i-1} + f_i$, and F_{-1} is defined as follows:

$$F_{-1} = \frac{1}{h^2}\left[y_{12} - \frac{1}{2}(y_{11} + y_{10}) - h^2(b_0 g_{11} + b_1 g_{10} + \cdots + b_{11} g_0)\right] \tag{13}$$

where

$$\begin{aligned} g_{11} &= f_{11} + g_{10} \\ g_{10} &= f_{10} + g_9 \\ &\vdots \\ g_1 &= f_1 + g_0 \\ g_0 &= f_0 \end{aligned}$$

Notice that transforming S3N5 into summed form, does not reduce the number of arithmetic operations in the position part of the formula. This is because the original S3N5 formula really involved 3 arithmetic operations, it is a three-point formula, with the a_1 coefficient equal to zero. Nevertheless, the summed form is very desirable in practice because the subtraction in the original formula has been replaced with the average of the two previous positions. I haven't tested this experimentally.

8 CONCLUSION AND OPEN PROBLEMS

The main points of my paper are as follows: First, it is the theory of error growth for multistep methods on smooth two-body orbits, including the numerous useful formulas regarding the error as a function of time, stepsize, orbital period, and eccentricity. Secondly, it is the search for new integrators, including the new method S3N5, and also the limitations of multistep methods regarding stability and order of accuracy.

With regard to future research, I have summarized below some of the most interesting problems, as I see them, relating to this work, that remain open.

The first problem is to investigate further the cancellation in angular error, which occurs for predictors of order 12, on the Sun-Jupiter system, stepsize around 32 days, double precision arithmetic (see figure 13 on page 93). Perhaps, my proposed spiral-in spiral-out idea can be developed to the point where it can predict, at least for certain cases, when cancellation will occur. My explanation that the spiral-in spiral-out behavior of the error is due to interactions between roundoff and truncation, is too vague to be useful.

Related to the preceding problem is the following: To find a multistep predictor method which tends to spiral inwards on the Sun-Jupiter problem, at a stepsize around 32 days. The idea is we could alternate such a method with a method like Stormer, that spirals outwards. Switching back and forth the two methods could be done, perhaps, every 100 revolutions. The effect would be to cause artificially and repeatedly a cancellation in angular error, much like the cancellation occurring at the initial stages of the integration for Stormer-12 (double-precision).

I must say that all the predictors I have tested (Stormer, S3N5, and many other others), eventually spiral outwards on the Sun-Jupiter problem, stepsize 32 days; that is, their energy error becomes positive. It seems that the spiral-outwards tendency is a property of the truncation error. This needs to be studied further

analytically, and perhaps a theorem can be found expressing which methods tend to spiral inwards and which outwards, and when.

Another pertinent experimental fact, is that the predictor-corrector Stormer-Cowell spirals inwards. I have seen this both for order 12 (see figure 5 on page 85), and for order 13. So perhaps, if we can't find any predictor that spirals inwards, *and* if we can afford the cost of a corrector, a thing to do is to alternate Stormer-Cowell with Stormer. The alternation must be done over a long time scale, say 100 orbits, to allow enough time for each integrator to develop its spiraling tendency. Perhaps, such a combination would result in an overall linear error growth in position, it would break the quadratic growth barrier!

Continuing with my list of open problems, the next one is to finish my proof for the truncation resonance in two-body systems of non-zero eccentricity. To do so, one may have to use perturbation expansions and other such techniques. The ultimate goal, though, should be to come up with a simple theory for the truncation resonances. Any complicated proof will be of little use, if it can not be translated into a simple physical argument, so that it can be used in practice.

Also, a second thing about the truncation resonance is the following conjecture: All methods based on polynomial approximation exhibit truncation resonance on the two-body problem. The t^2 error of Runge-Kutta-4 is a good sign. A first step would be to argue that all polynomial methods exhibit truncation resonance on the harmonic oscillator. This shouldn't be too hard. The local truncation error is, more or less, the first omitted term of a Taylor series, and oscillates like the true solution. The accumulated error must also follow the growth and decay of the true solution (should it?). Then, it follows that a linear truncation resonance must take place. More experimental evidence and more analytic work are needed.

One more interesting project that springs off this research, is to investigate other nonlinear oscillations, in addition to the two-body problem. Perhaps, one

can find a problem that has a different angular-shear law than the two-body. For example, an interesting shear law would be one that integrates a one-time error in radius by t^2 as opposed to t, as the two-body does. Then, if my theory of error is any good, and if nothing unexpected happens in the new situation, the truncation error in the new problem should grow like t^3.

The final project I have in mind, is to investigate the limitation of multistep methods between order of accuracy, stability, and precision size needed to represent exactly the b_i coefficients of the method. As noted previously, the stability boundary on the harmonic oscillator shrinks as the order of accuracy of a method increases. The precision size needed to represent exactly the b_i coefficients increases, as the order of accuracy of a method increases. Why is that so? There's probably something deep behind this, something about polynomial approximation. And then, the completely unstable method H615 has coefficients that require very small precision to be represented. That also needs explanation, unless we want to label it a coincidence.

References

[1] H. ABELSON, G. & J. SUSSMAN: *Structure and Interpretation of Computer Programs*, MIT Press, 1986.

[2] C. COHEN, E. HUBBARD, C. OESTERWINTER: "Elements of the outer planets for one million years," *Astronomical Papers of the American Ephemeris*, XXII, I, 1973.

[3] T.J. DEKKER: "A floating-point technique for extending the available precision," *Numer. Math. 18*, 224-242, 1971.

[4] R.W. HAMMING: *Numerical Methods for Scientists and Engineers*, Dover, 1986.

[5] P. HENRICI: *Discrete Variable Methods in Ordinary Differential Equations*, John Wiley & Sons Inc., 1962.

[6] P. HENRICI: *Error Propagation for Difference Methods*, John Wiley & Sons Inc., 1963.

[7] H. KINOSHITA, H. NAKAI: "Stability of PLuto's orbit and orbital motions of outer planets for five million years" in *Tokyo Astronomical Observatory*, 1-21-1.

[8] S.W. McCUSKEY: *Introduction to Celestial Mechanics*, Addison-Wesley, 1963.

[9] J. APPLEGATE, M. DOUGLAS, Y. GURSEL, G. SUSSMAN, J. WISDOM: "The Outer Solar System for 200 Million Years," in *Lecture Notes in Physics #267* – Use of supercomputers in Stellar Dynamics, Springer Verlag, 1986. Reprint of Astron. J. article.

[10] E. STIEFEL, G. SCHEIFELE: *Linear and Regular Celestial Mechanics*, Springer-Vergal, 1971.

[11] E. STIEFEL, D. BETTIS: "Stabilization of cowell's method," *Numer. Math. 13*, p. 154-175, 1969.

9 APPENDIX

The first item in this appendix is a listing of a program that computes the coefficients of multistep predictor methods of the general form

$$y_{n+1} = a_0 y_n + a_1 y_{n-1} + \cdots + h^2 [b_0 f(y_n) + b_1 f(y_{n-1}) + \cdots + b_k f(y_{n-k})]$$

The program is complete except for the implementation of rational arithmetic and a few polynomial manipulation procedures not shown for lack of space. I will be happy to provide them to anyone who asks me. The main routine is `gamma` and takes one argument, a list of coefficients (a0 a1 a2 a3 ...), and returns a stream of gamma coefficients. If we wish to construct a 12th order method, we simply evaluate only the first 12 elements of the resulting stream. The procedure `gamma` implements the recursion relation 6 given in Section 5, and uses the procedure `left-expansion` to compute $\rho_0, \ldots, \rho_m$. The procedure `left-expansion` performs the expansion in ζ by doing long division on the result of `rho-in-zeta-1`, which computes the polynomial $\rho(z)(1-\zeta)$ in powers of ζ. The rest of the code is sub-procedures which should be self-explanatory.

```
(define (rho-in-zeta-1 rho-p)  ; Return the polynomial = rho(z) * (1-zeta)
  ;; rho-p is the list of coefficients '(a0 a1 a2 a3 ...)
  (define ncoefs (length rho-p))
  (define rho-factors (cons 1 (map - rho-p)))
  (define rho-powers  (cons 0 (substream->list the-integers
                                               0 (-1+ ncoefs))))
  (define polys (map scale-poly
                     rho-factors
                     (map 1-z^n-poly rho-powers)))
  (define total-p (apply (accumulation add-polys (list 0)) polys))
  total-p)

(define (left-expansion rho-p)   ; Return stream of Left-Hand-Side terms
  (let ((p (rho-in-zeta-1 rho-p)))
    (if (and (= (first p) 0) (= (second p) 0)) ; Divisible by zeta^2
        (error "coefs fail to satisfy y=1,x" rho-p)
        (inverse-long-division (cddr p) (list 1 -1)))))
```

```
(define (gammas rho-p)      ; Return stream of gamma coefficients
  (define LHS (left-expansion rho-p))
  (define the-gammas
    (cons-stream
     (head LHS)
     (map-2-streams
      (lambda (rho_m m)
        (- rho_m
           (vector-sum
            (vector-mul
             (substream->vector the-2/m+2 1 m)
             (vector-mul (harmonic-sums->vector 2 (1+ m))
                         (vector-reverse
                          (substream->vector the-gammas
                                             0 (-1+ m))))))))
      (tail LHS) the-integers)))
  the-gammas)


(define (gamma->beta gamma k i)
  ;; gamma= stream of difference-form coefficients
  ;; k    = use up to (k-1) difference, gives k beta coeffs
  ;; i    = compute the ith beta coefficient, 0=<i<k
  (* (expt -1 i)
     (vector-sum
      (vector-mul (substream->vector gamma i (- k 1))
                  (substream->vector (binomials_ij_j=i0 i)
                                     0
                                     (- (- k 1) i))))))


;; SUB-PROCEDURES


(define (1-z^n-poly n)        ; Return polynomial (1-z)^n
  (poly-arg-scale (stream-ref pascal-triangle n) -1))

(define pascal-triangle
  (cons-stream
   (list 1)
   (map-stream (lambda (previous)
                 (map + (cons 0 previous) (append previous '(0))))
               pascal-triangle)))
```

```
(define (inverse-long-division num den)  ;   Return stream of terms
  (let ((r (/ (first num) (first den)))) ;   ratio-of-first-terms
    (cons-stream
     r
     (inverse-long-division (sub-polys (cdr num)
                                       (scale-poly r (cdr den)))
                            den))))

(define the-integers                     ; 1 2 3
  (cons-stream 1 (map-stream 1+ the-integers)))

(define the-2/m+2                        ; 2/2, 2/3, 2/4, 2/5, ...
  (map-stream (lambda (m+2) (/ 2 m+2))
              (tail the-integers)))

(define the-harmonic-sums                ; Nth partial sum n=1,2,3, ...
  (let ((the-inverse-integers (map-stream / the-integers)))
    (stream-accumulate the-inverse-integers + 0)))

(define (harmonic-sums->vector from to) ; from=2 gives 2nd harmonic-sum
  (if (or (< from 1) (< to from))
      (error "harmonic-sums: illegal-indices---" from to)
      (substream->vector the-harmonic-sums (-1+ from) (-1+ to))))

(define (binomials_ij_j=i0 i)            ; (i i),(i+1 i),(i+2 i), ...
  ;; recursion formula: [choose j+1 i] = (1+j)/(1+j-i) * [choose j i]
  (define the-binomials
    (if (= i 0)
        (cons-stream 1 the-binomials)
        (cons-stream
         (rat 1)
         (map-2-streams (lambda (j binomial-of-j)
                          (* (/ (1+ j) (1+ (- j i)))
                             binomial-of-j))
                        (stream-cdr the-integers (-1+ i))
                        the-binomials))))
  the-binomials)
```

COEFFICIENTS FOR S3N5

I have listed below the γ_i difference-form coefficients, and the b_i standard-form coefficients for the predictor S3N5. The difference-form and standard-form equations are as follows:

$$y_{n+1} = \tfrac{1}{2}(3y_n - y_{n-2}) + h^2\left[\gamma_0\Delta^0 f(y_n) + \gamma_1\Delta^1 f(y_n) + \cdots + \gamma_k\Delta^k f(y_n)\right]$$

$$y_{n+1} = \tfrac{1}{2}(3y_n - y_{n-2}) + h^2\left[b_0 f(y_n) + b_1 f(y_{n-1}) + \cdots + b_k f(y_{n-k})\right]$$

The summed form equation, given in page 70, uses the same b_i coefficients as the standard form.

i	γ_i coefficient	γ_i/γ_0
0	3 / 2	1
1	-1 / 2	-.33
2	1 / 8	8.3e-2
3	1 / 12	5.6e-2
4	37 / 480	5.1e-2
5	7 / 96	4.9e-2
6	2803 / 40320	4.6e-2
7	403 / 6048	4.4e-2
8	155171 / 2419200	4.3e-2
9	64243 / 1036800	4.1e-2
10	19172441 / 319334400	4.0e-2
11	443453 / 7603200	3.9e-2
12	99008658523 / 1743565824000	3.8e-2
13	26340441241 / 475517952000	3.7e-2
14	1132479023419 / 20922789888000	3.6e-2
15	31968233083 / 603542016000	3.5e-2

I have listed the b_i coefficients between orders 2 and 15. They are expressed in numerator-denominator form, with the numerators listed in the order $b_0, b_1, b_2, \ldots$,

and the common denominator listed as the last number in each group. Note that the coefficients for orders of accuracy higher than 14, do not fit in 53 bits mantissa.

Order 2: [9, 2, 1, 8]

Order 3: [29, 0, 9, -2, 24]

Order 4: [617, -148, 402, -188, 37, 480]

Order 5: [652, -323, 752, -538, 212, -35, 480]

Order 6: [57571, -43950, 105213, -101252, 59853, -19758, 2803, 40320]

Order 7: [180773, -188270, 484899, -585856, 461659, -228534, 64829, -8060, 120960]

Order 8: [3770631, -5006768, 14042768, -20406696, 20095150, -13260256, 5641368, -1402568, 155171, 2419200]

Order 9: [11761594, -19067613, 58317540, -98994972, 116947776, -96443094, 54698988, -20396940, 4512822, -449701, 7257600]

Order 10: [536682577, -1030699382, 3428731605, -6656471688, 9171914754, -9074951268, 6432968082, -3198158280, 1061324013, -211511254, 19172441, 319334400]

Order 11: [555307603, -1235574668, 4453108035, -9729600978, 15318173334, -17679713280, 15037730094, -9344416860, 4134453303, -1235887684, 224047727, -18625026, 319334400]

Order 12: [3130988170903, -7934341589556, 30848541333618, -74905526214940, 132646512372525, -174946092059016, 173590006788492, -129435373605816, 71583401003265, -28529851629700, 7757872051938, -1289796544236, 99008658523, 1743565824000]

Order 13: [9682709366360, -27569707866131, 115145722585632, -307583606789006, 605107107478040, -897739902825885, 1017972189230592, -885508289682564, 587651829658632, -292757125249565, 106140644300000, -26469488217486, 4063709073032, -289744853651, 5230697472000]

Order 14: [39863316488859, -126133537792390, 563638481473657, -1642556791680540, 3554039932354579, -5858182616188378, 7472723264249625, -7428701167104264, 5751441825961785, -3438251505883098, 1558174079642419, -518100317394460, 119310427423257, -17013685742470, 1132479023419, 20922789888000]

Order 15: [122914645707209, -428271056986650, 2040008549687331, -6440407164529180, 15200330165526417, -27558610659183030, 39058274477112035, -43[illegible]779712, 38648745786352275, -26954859202012454, 14658585049545153, -6092511320646060, 1870668071757331, -400134162493770, 53267880679737, -[illegible], 62768369664000]

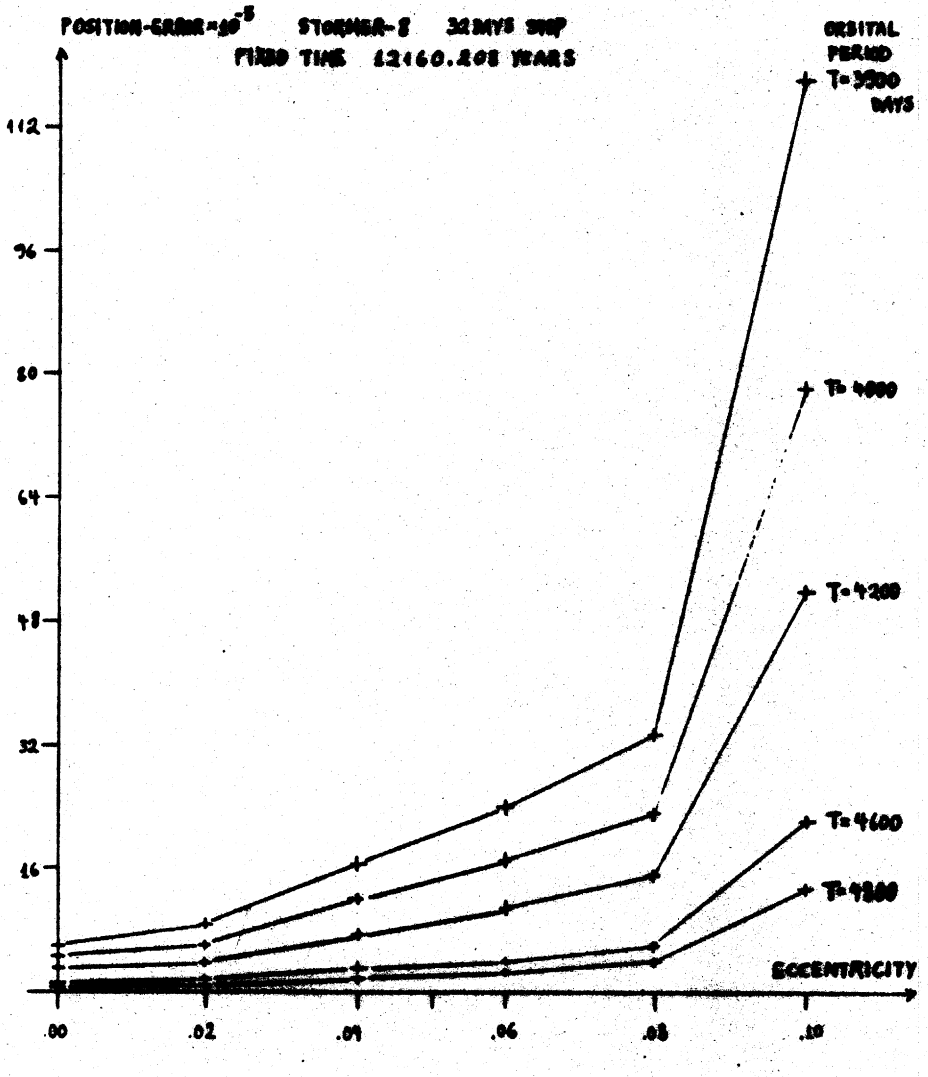


Figure 1: Error vs. period for different values of eccentricity.

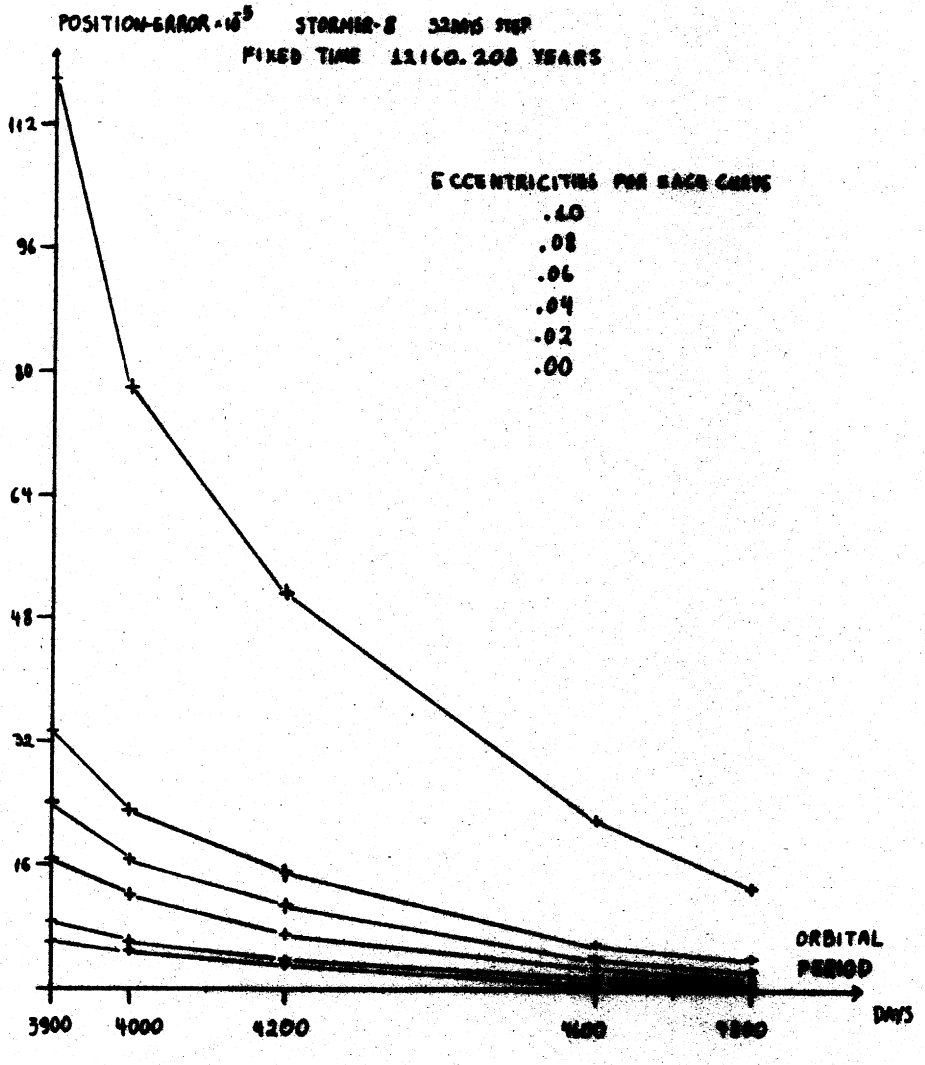


Figure 2: Error vs. eccentricity for different values of period.

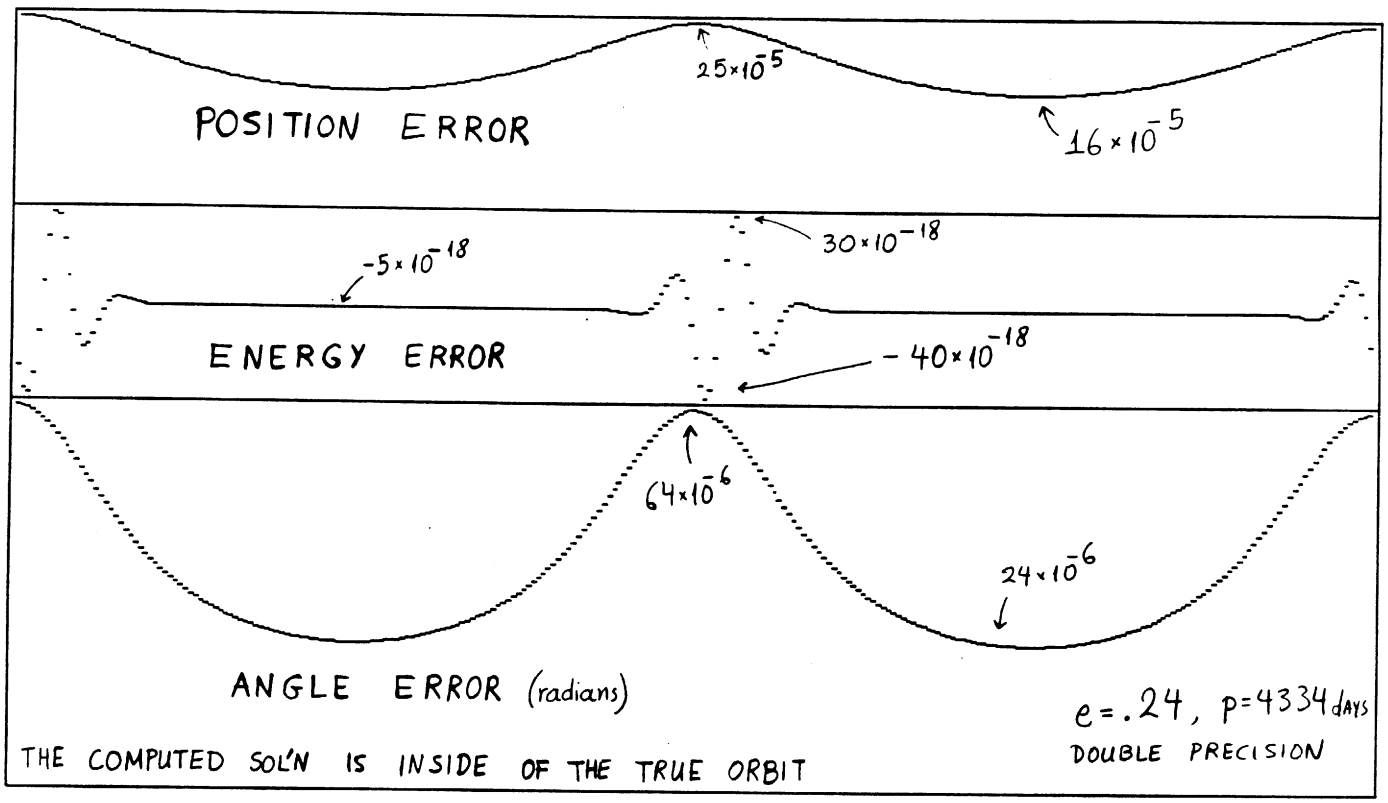


Figure 3: Error in Position, Energy, and Angle at eccentricity .24, shown during 2 orbits after 16x1024 revolutions, orbital period 4334 days, Stormer-12, stepsize 32 days.

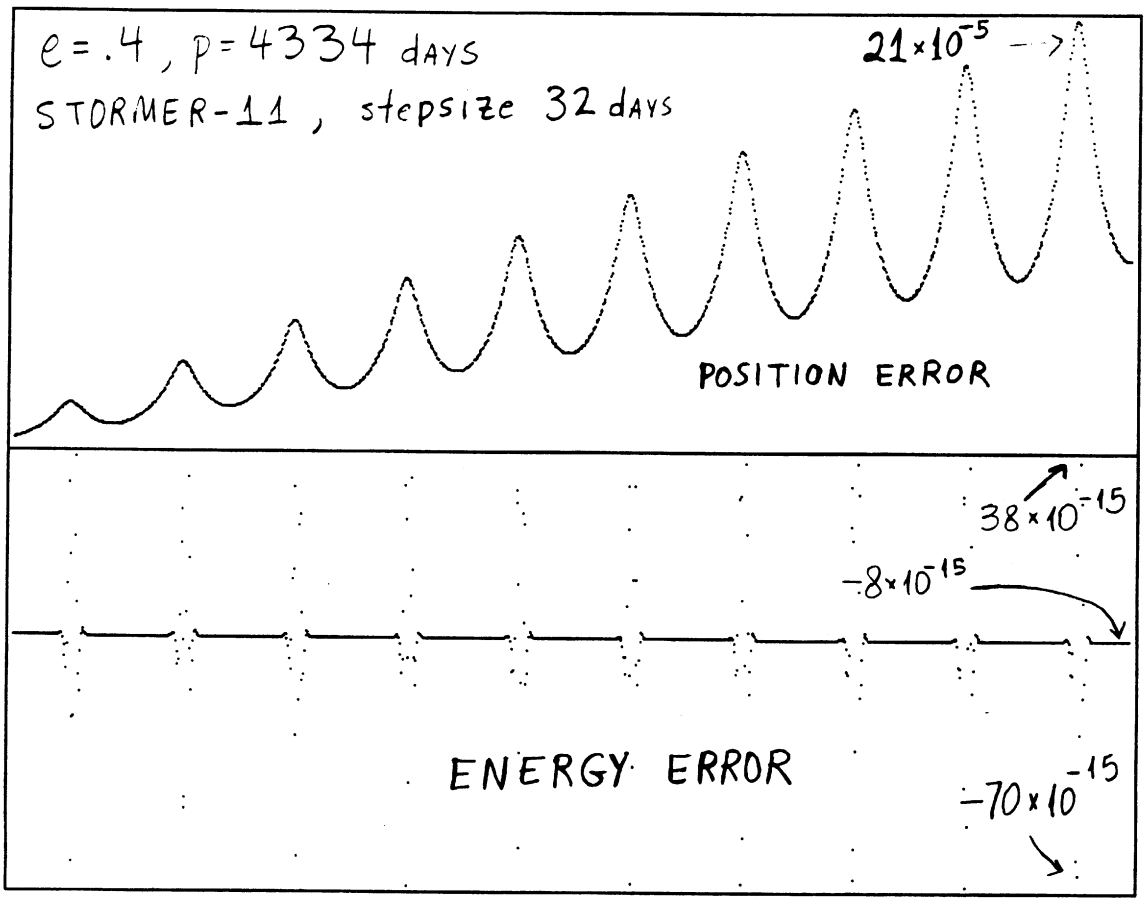


Figure 4: Error in Position and Energy at eccentricity .4, shown during first 10 revolutions, orbital period 4334 days, Stormer-11, stepsize 32 days.

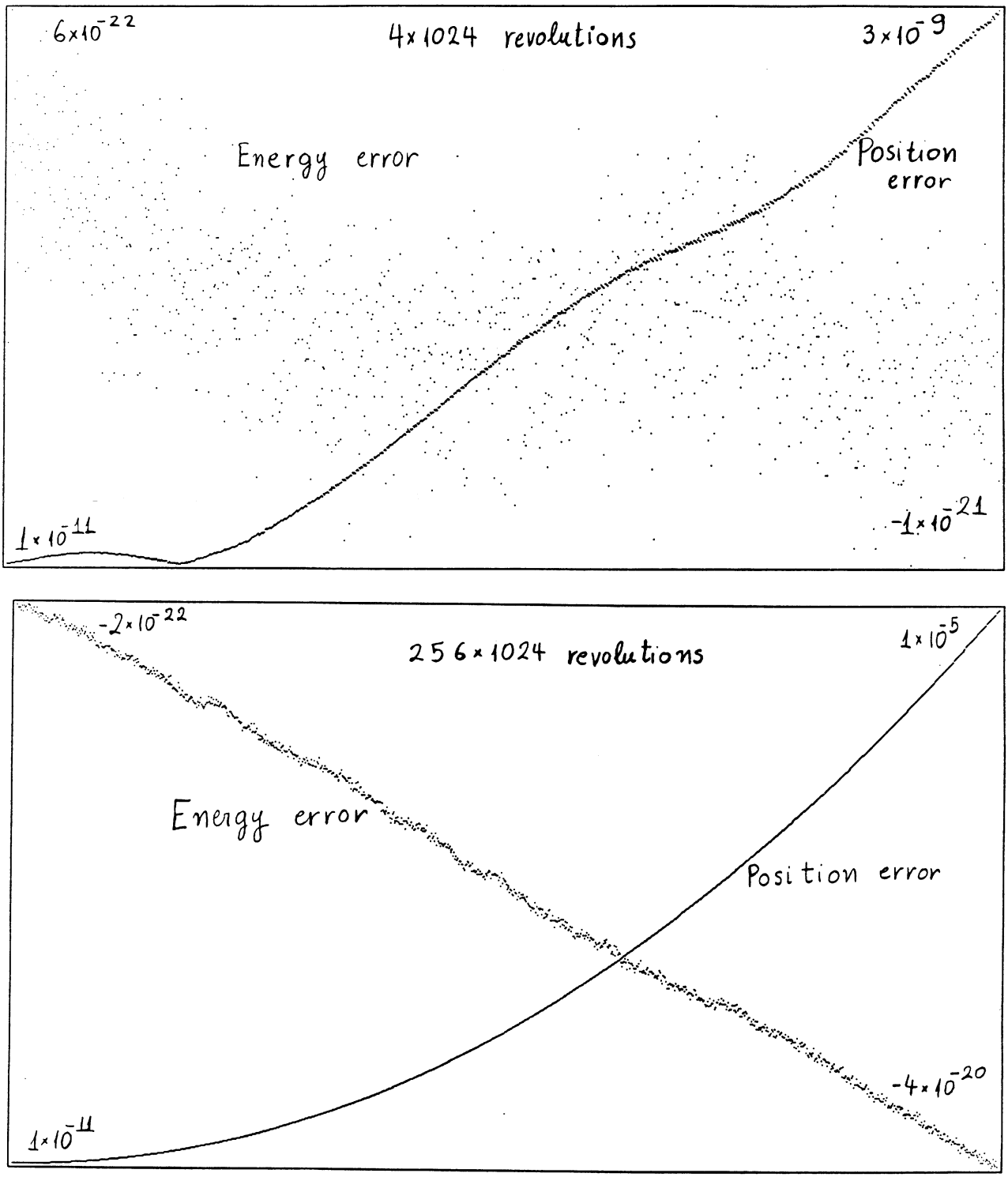


Figure 5: Error in Position and Energy for Stormer-Cowell-12 on the Sun-Jupiter system, stepsize 32 days, (quad precision).

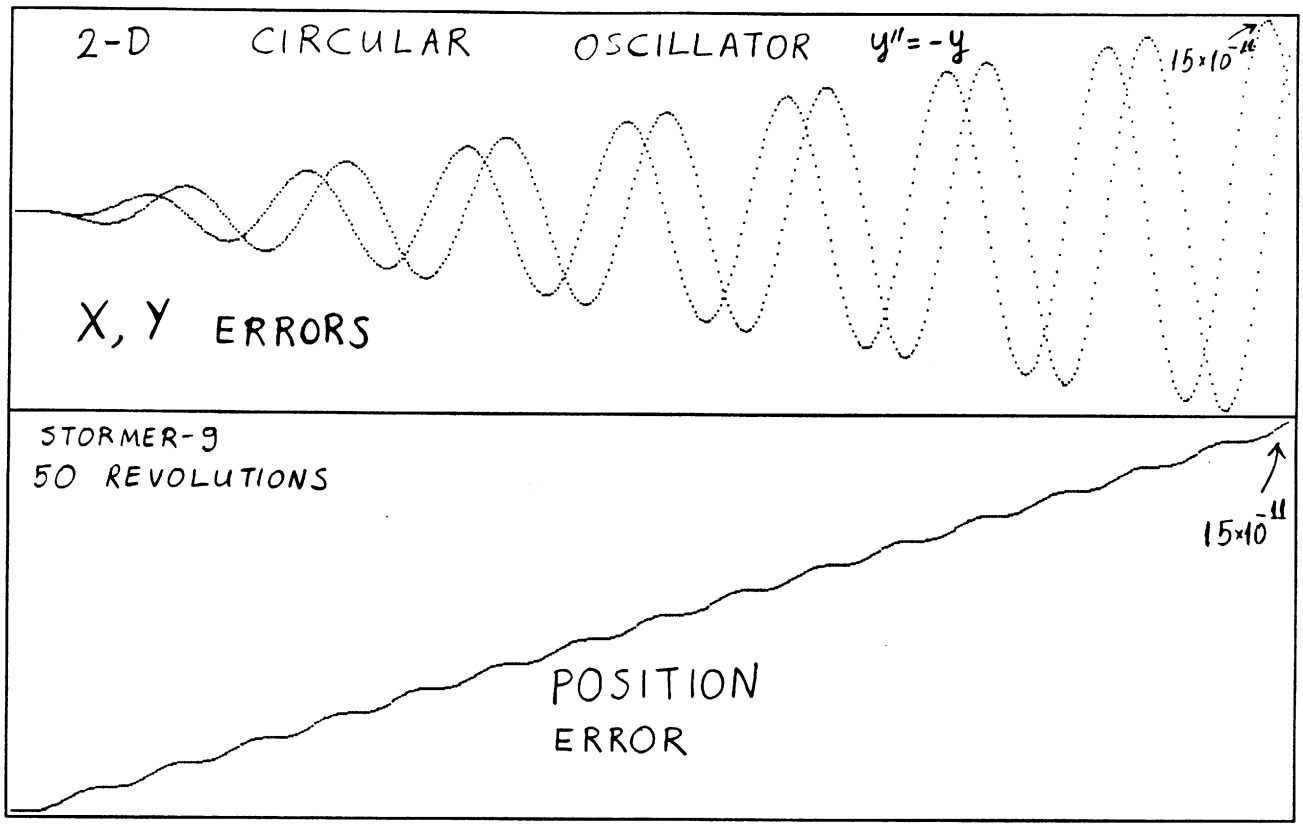


Figure 6: Error in X, Y, and Position on a two-dimensional *circular* oscillator $y'' = -y$. Shown during first 50 revolutions, stepsize .1, initial conditions $[x, y, x', y'] = [1, 0, 0, 1]$, Stormer-9.

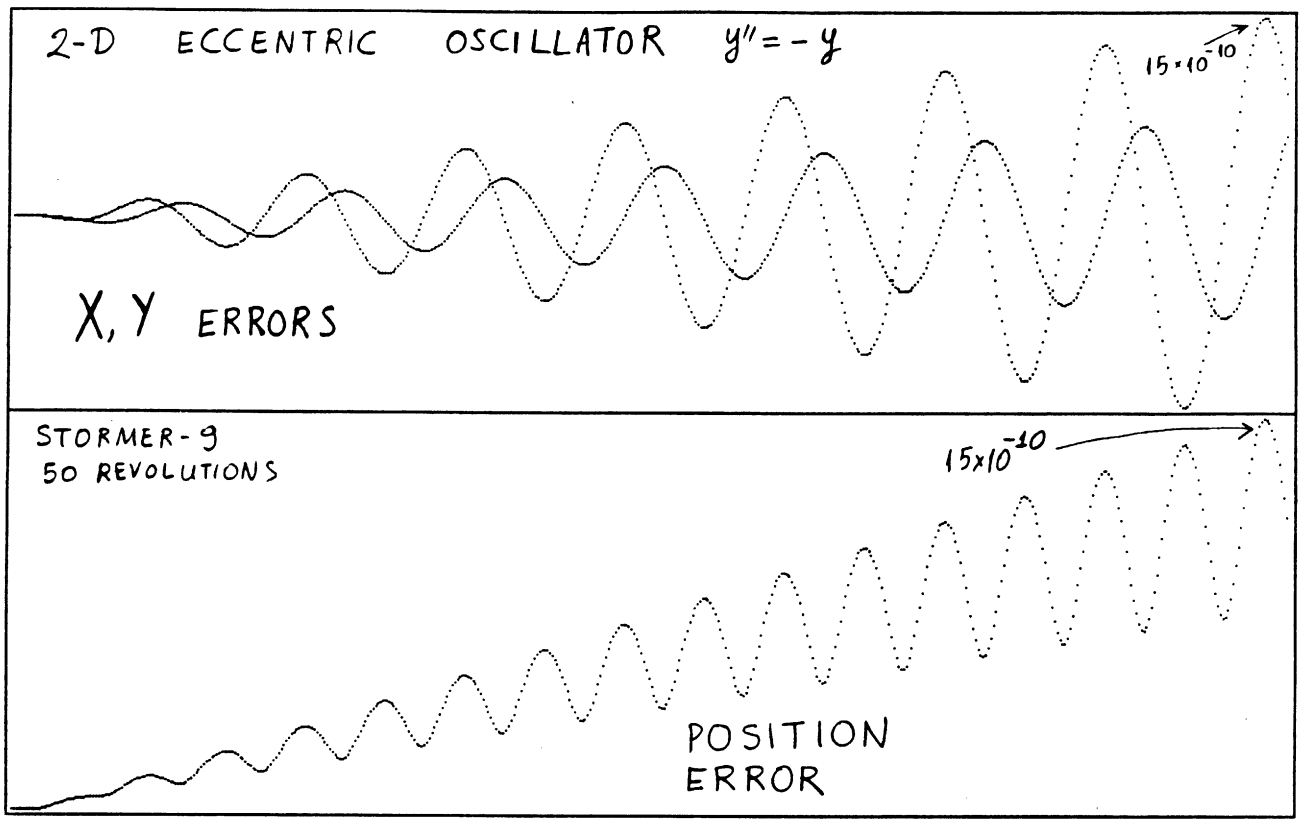


Figure 7: Error in X,Y, and Position on a two-dimensional *eccentric* oscillator $y'' = -y$. Shown during first 50 revolutions, stepsize .1, initial conditions $[x, y, x', y'] = [1, 0, 0, .5]$, Stormer-9.

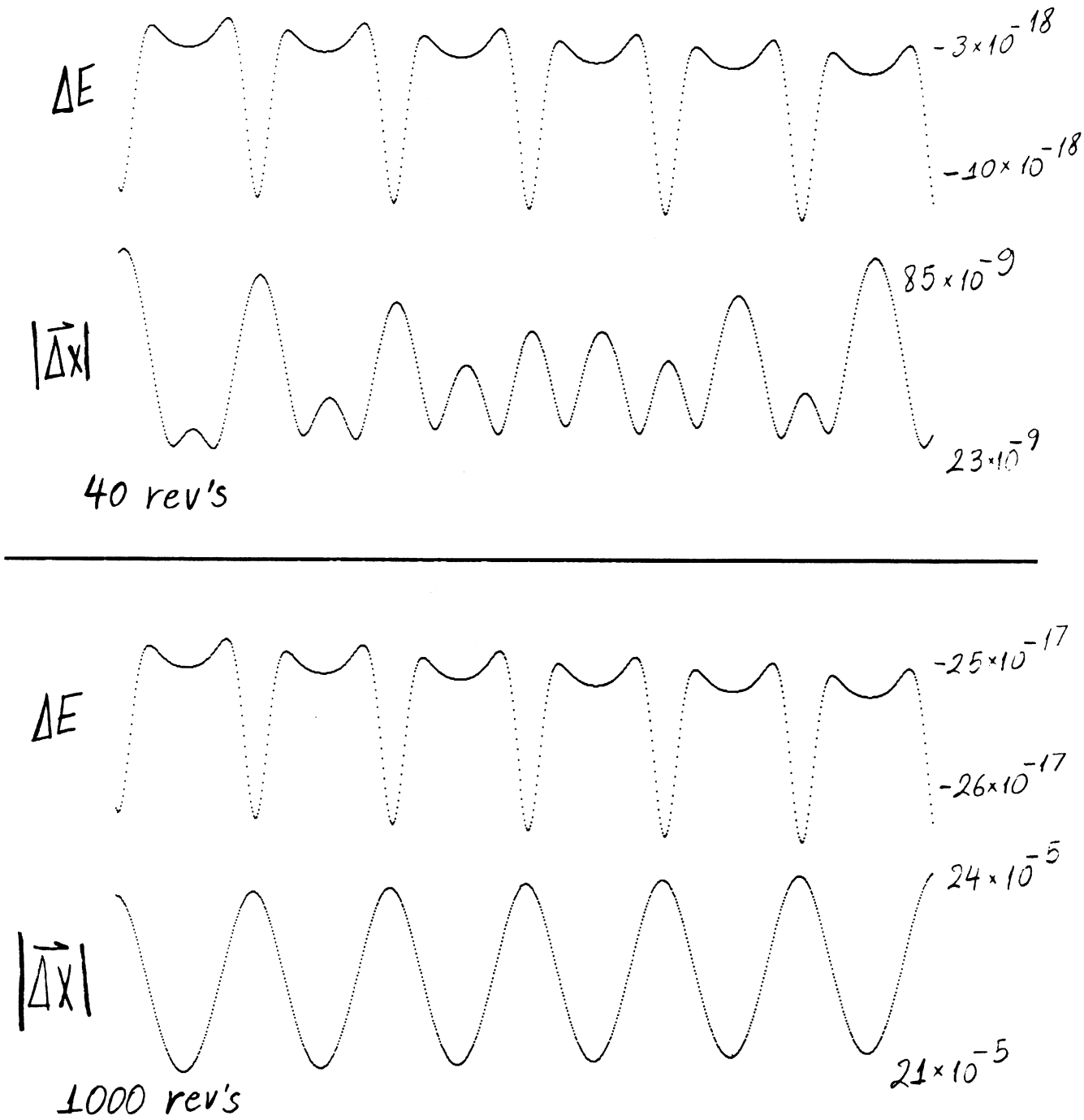


Figure 8: Error in Position and Energy vs. time for Stormer-8 on the Sun-Jupiter system, stepsize 32 days. The time axis on the two top graphs runs from 40 to 46 revolutions, and on the bottom two graphs from 1000 to 1006 revolutions.

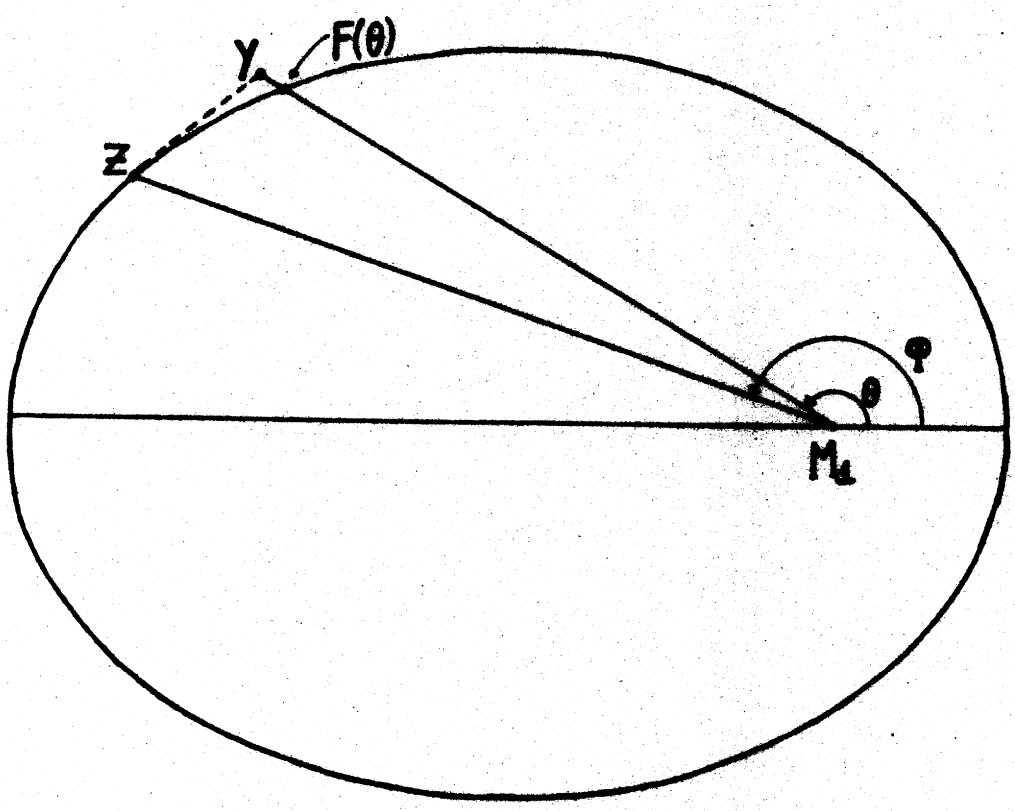


Figure 9: The distinction between Radial and Angular Error.

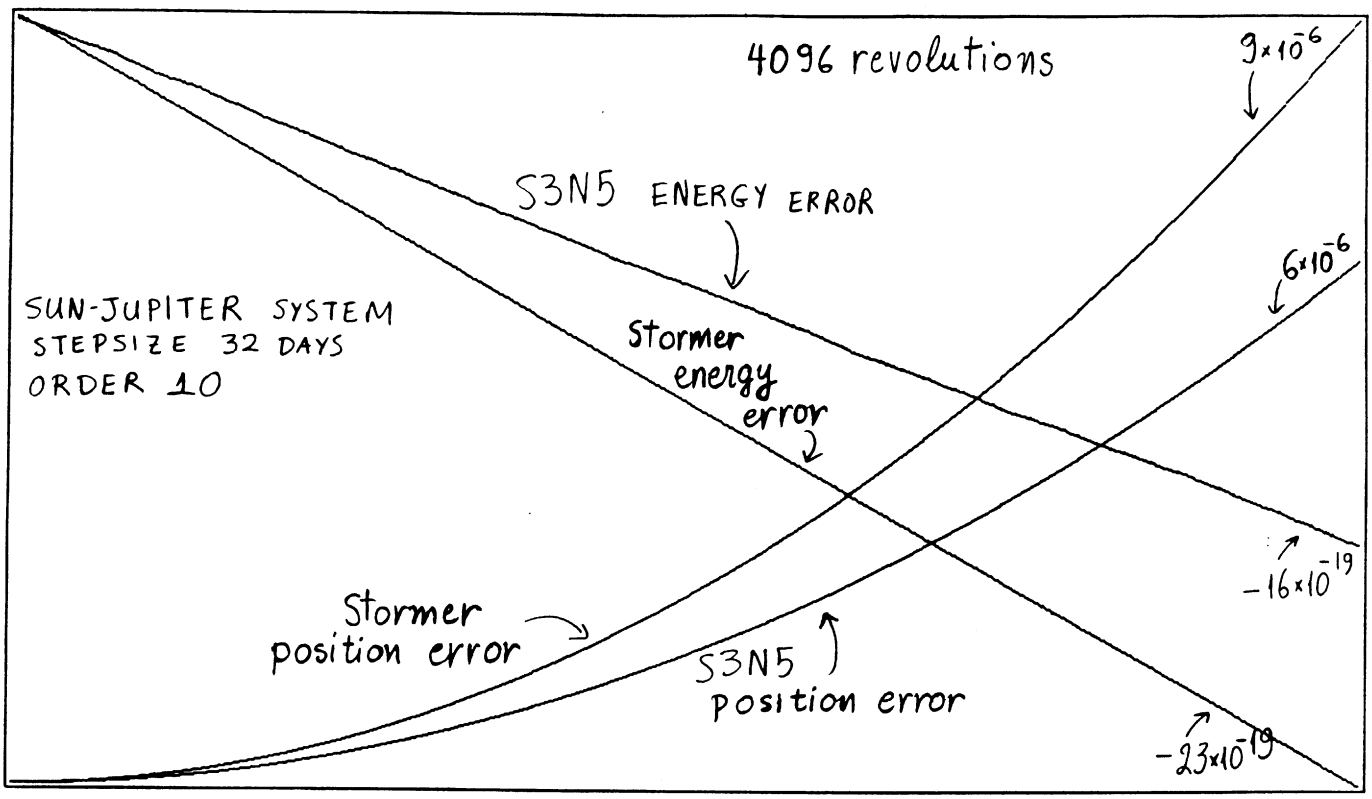


Figure 10: Error in Position, and Energy for Stormer-10 and S3N5-10, on the Sun-Jupiter system. Time runs from 0 to 4096 revolutions.

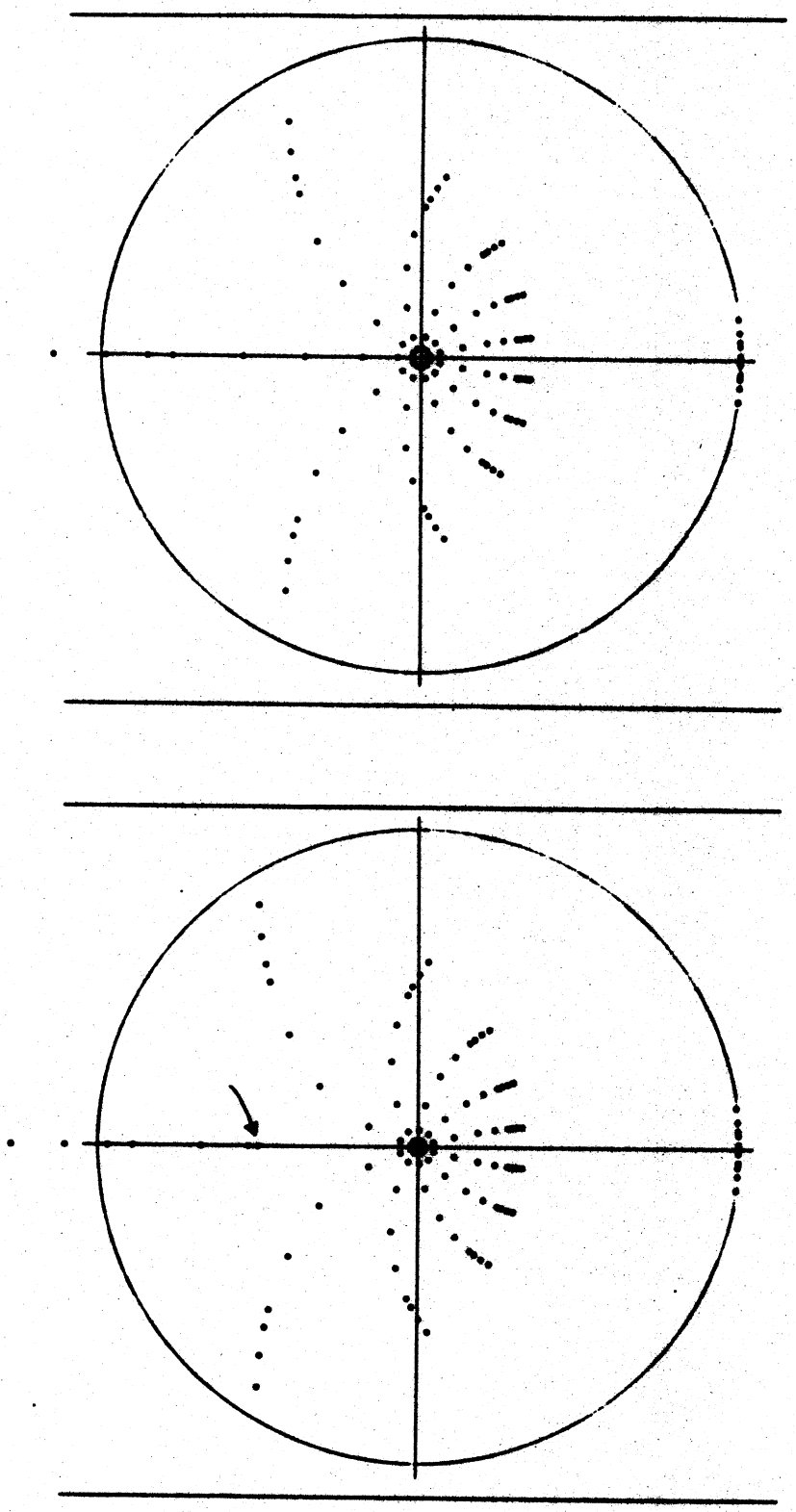
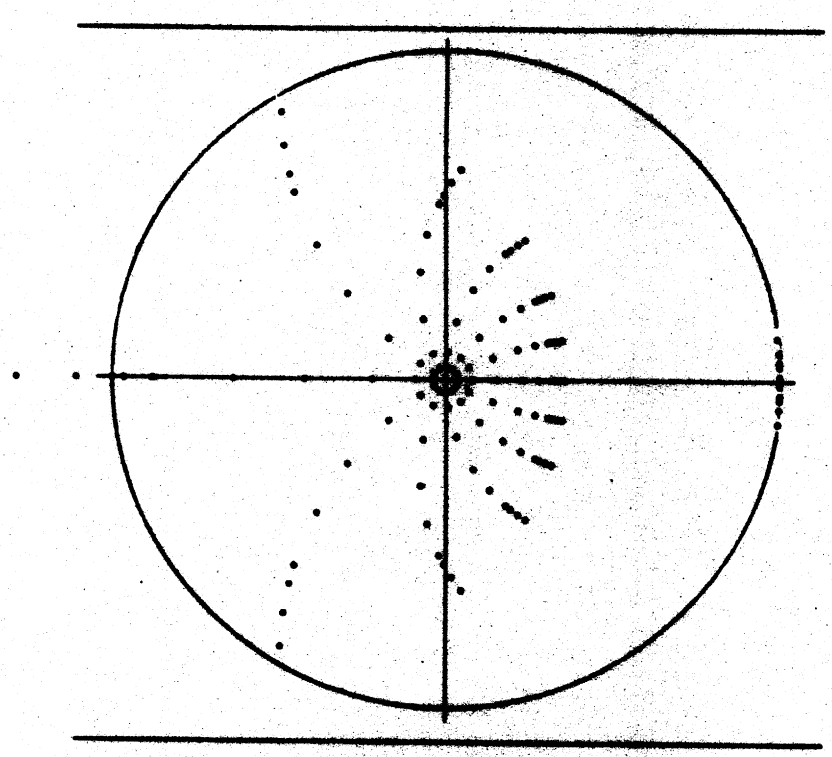


Figure 11: Spider plots for Stormer-12 and S2N5-12.



STORMER13, R=

Figure 12: Spider plot for Stormer-13.

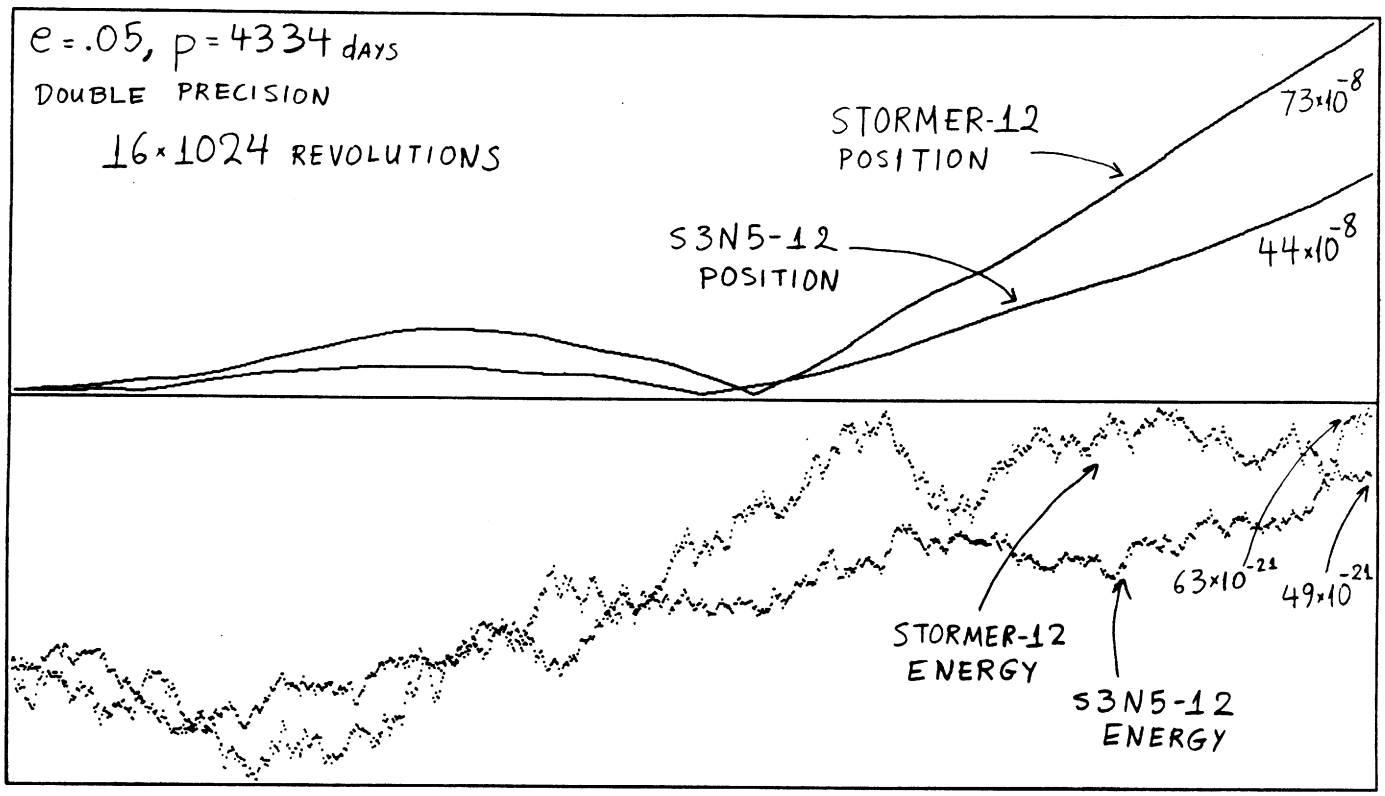


Figure 13: Error in Position and Energy for Stormer-12 and S3N5-12 — Double Precision — 16x1024 revolutions.

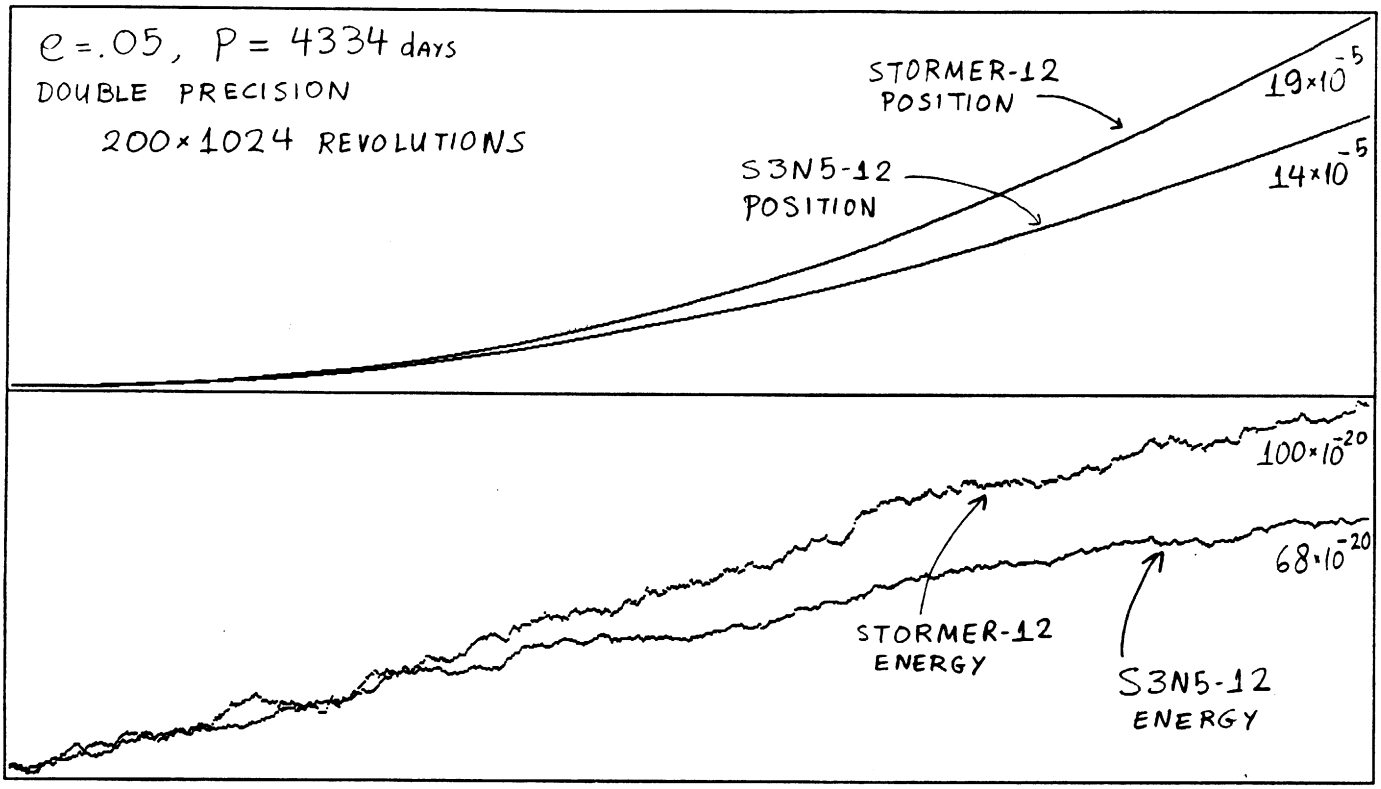


Figure 14: Error in Position and Energy for Stormer-12 and S3N5-12 — Double Precision — 200x1024 revolutions.

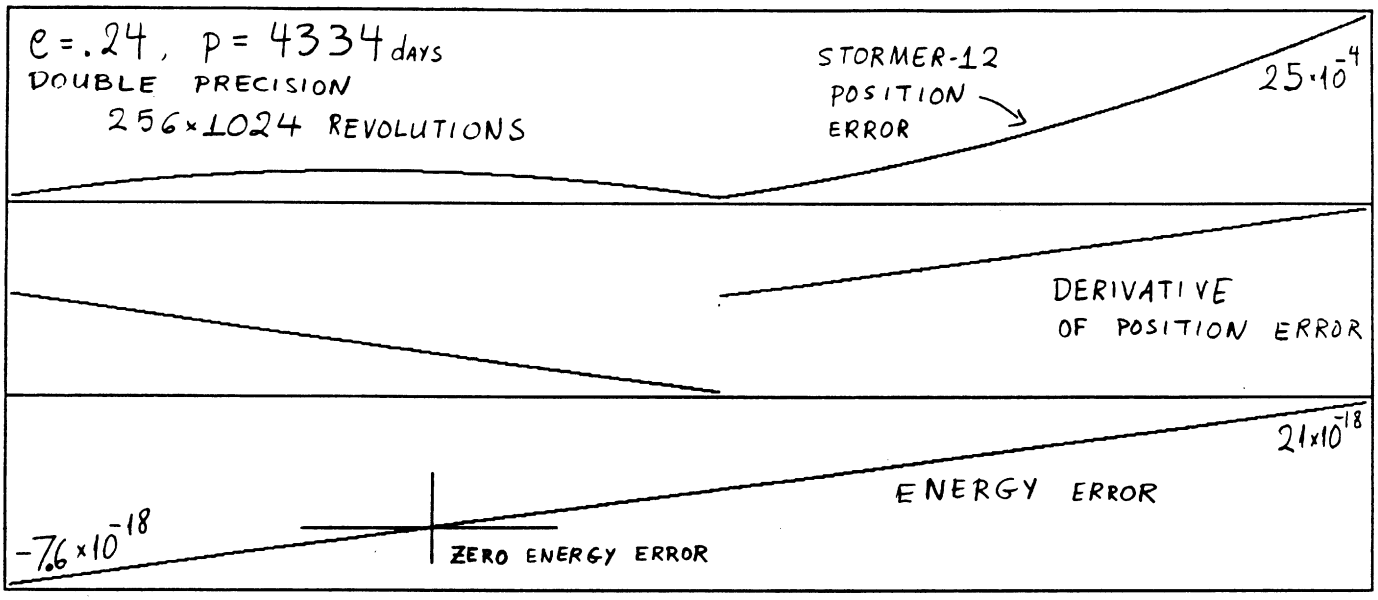


Figure 15: Peculiar cancellation in Position error: eccentricity .25, orbital period 4334 days, Stormer-12, stepsize 32 days, Double Precision.

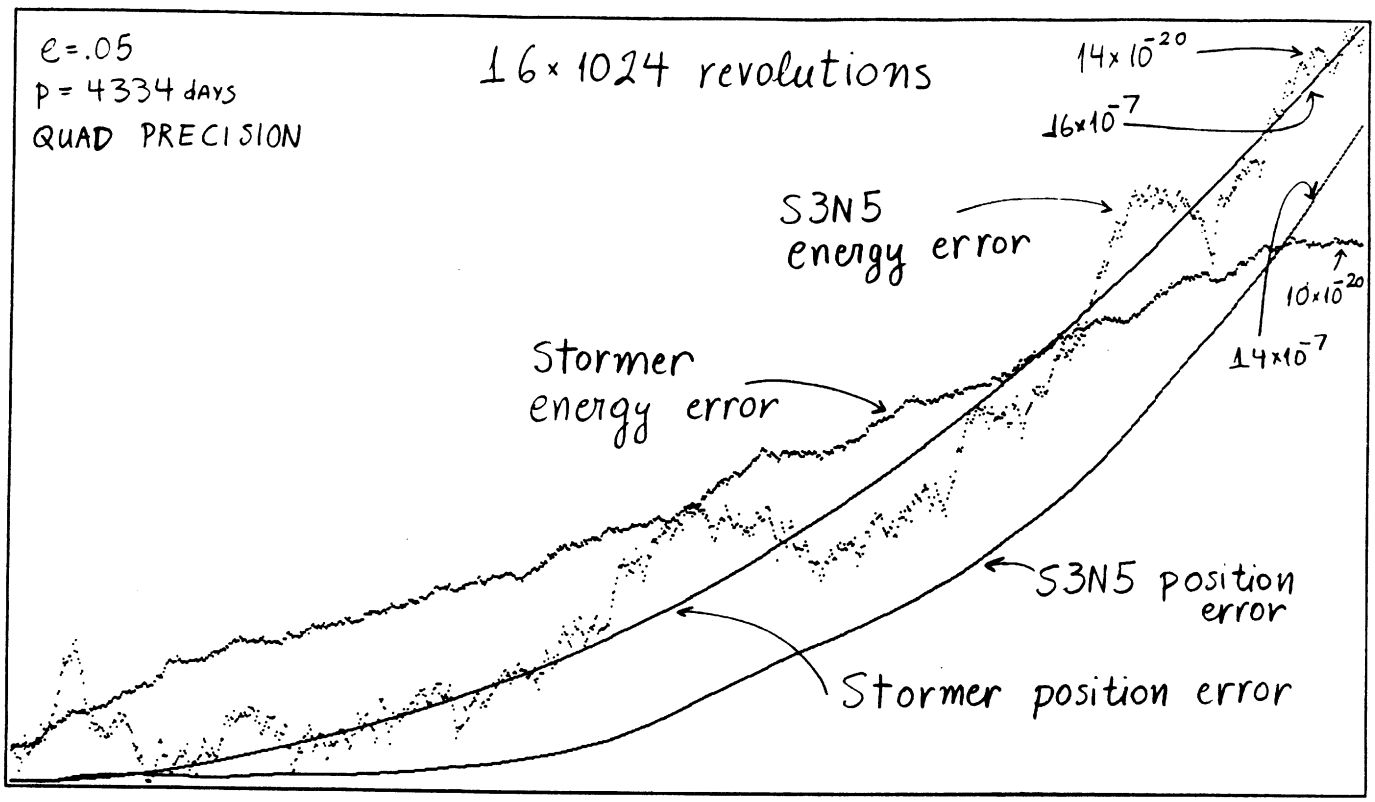


Figure 16: Error in Position and Energy for Stormer-12 and S3N5-12 — Quad Precision — 16x1024 revolutions.

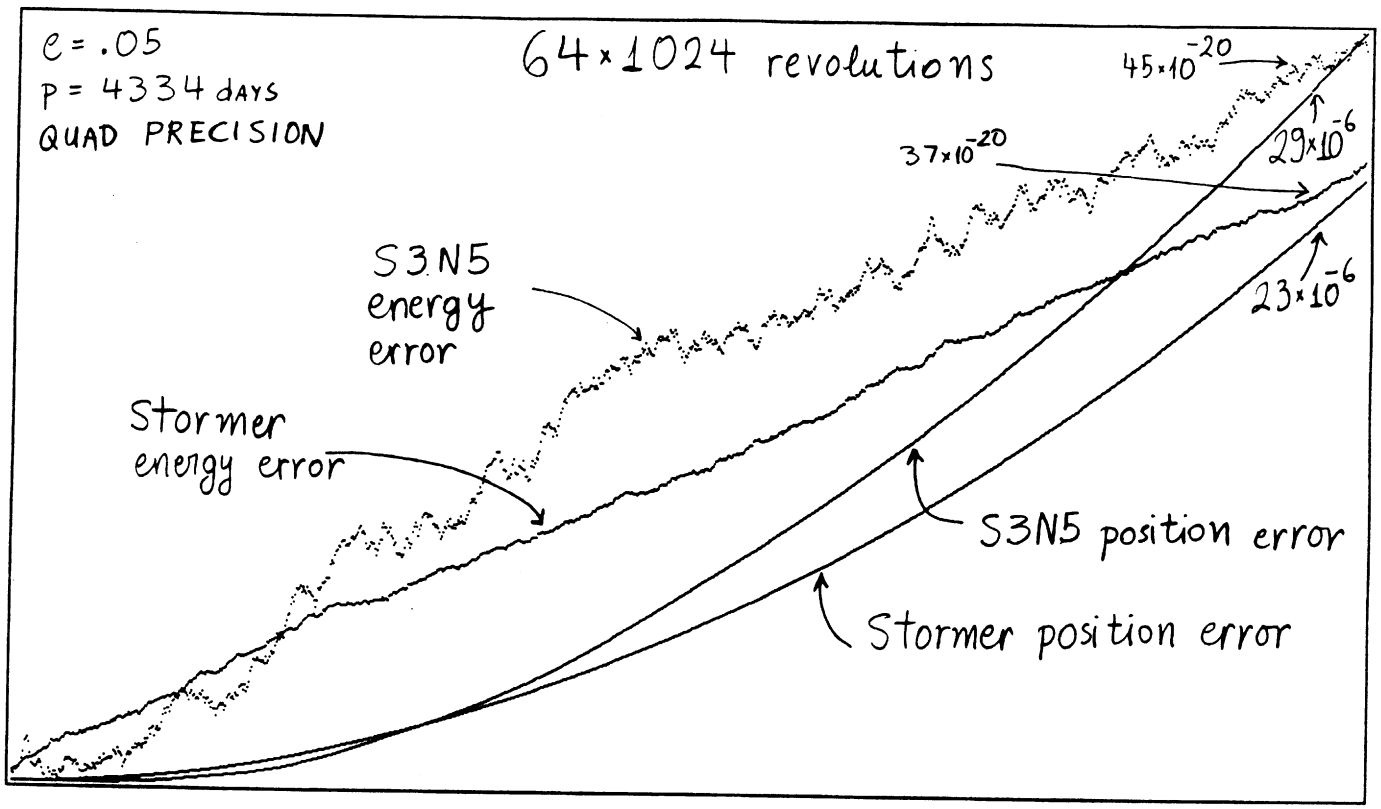


Figure 17: Error in Position and Energy for Stormer-12 and S3N5-12 — Quad Precision — 64x1024 revolutions.

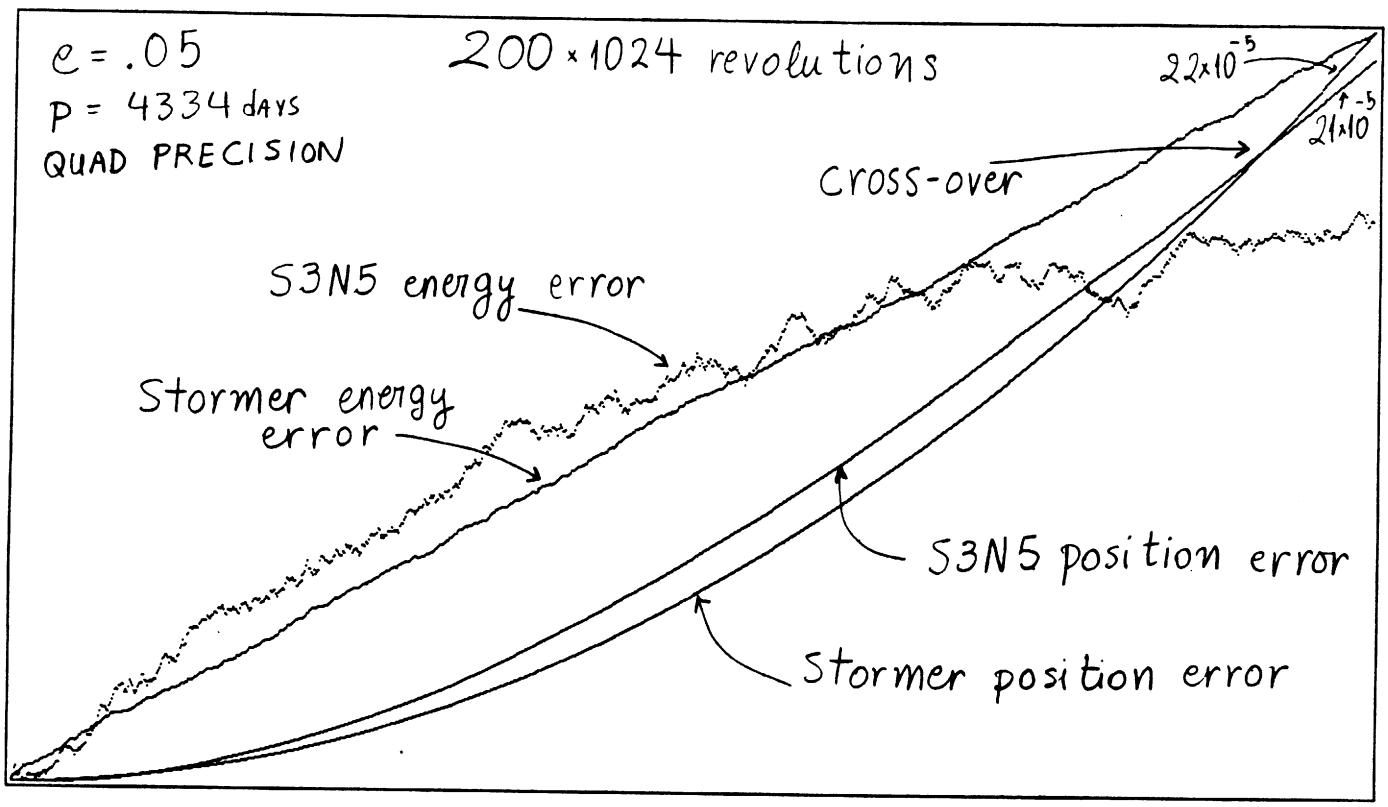


Figure 18: Error in Position and Energy for Stormer-12 and S3N5-12 — Quad Precision — 200x1024 revolutions.

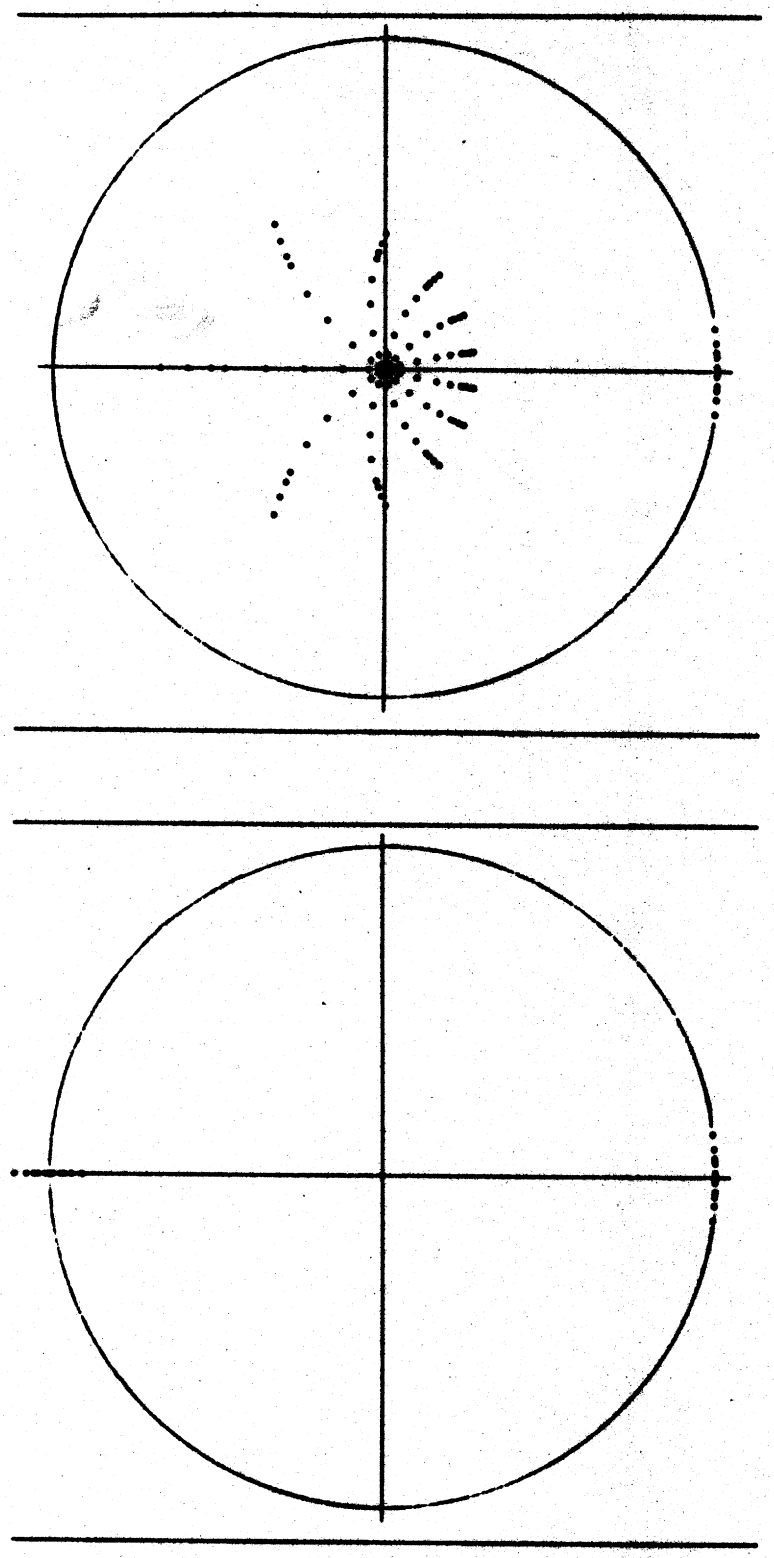


Figure 19: Spider plots for Cowell-13 and H615-4.

This blank page was inserted to preserve pagination.

CS-TR Scanning Project
Document Control Form

Date : 8 / 17 / 95

Report # AI-TR-1055

Each of the following should be identified by a checkmark:
Originating Department:

☒ Artificial Intellegence Laboratory (AI)
☐ Laboratory for Computer Science (LCS)

Document Type:

☒ Technical Report (TR) ☐ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 99 (107 - IMAGES)
Not to include DOD forms, printer instructions, etc... original pages only.

Originals are:
☒ Single-sided or
☐ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☒ Laser Print
☐ InkJet Printer ☐ Unknown ☐ Other:_____

Check each if included with document:

☒ DOD Form (2) ☐ Funding Agent Form ☒ Cover Page
☒ Spine ☐ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages (by page number):_____

Photographs/Tonal Material (by page number):_____

Other (note description/page number):

Description : Page Number:
IMAGE MAPS (1-99) PAGES #'ED 1-99 (INCLUDING TITLE PAGES)
(100-104) SCANCONTROL, COVER, SPINE, DOD(2)
(105-107) TRGT'S (3)

Scanning Agent Signoff:
Date Received: 8/17/95 Date Scanned: 8/23/95 Date Returned: 8/24/95

Scanning Agent Signature: Michael W. Cook

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AI-TR 1055	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): Multistep Methods for Integrating the Solar System	5. TYPE OF REPORT & PERIOD COVERED: technical report
6. PERFORMING ORG. REPORT NUMBER	
7. AUTHOR(s): Panayotis A. Skordos	8. CONTRACT OR GRANT NUMBER(s): N00014-86-K-0180
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Artificial Inteligence Laboratory, 545 Technology Square, Cambridge, MA 02139	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Advanced Research Projects Agency, 1400 Wilson Blvd., Arlington, VA 22209	12. REPORT DATE: July 1988
	13. NUMBER OF PAGES: 101
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Office of Naval Research, Information Systems, Arlington, VA 22217	15. SECURITY CLASS. (of this report)
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

numerical integration
solar system
two-body problem
error analysis
multistep analysis
roundoff error

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

High order multistep methods, run at constant stepsize, are one of the most effective schemes for integrating the Newtonian solar system, for extended periods of time. I have studied the stability and error growth of these methods, when applied to harmonic oscillators and two-body systems like the Sun-Jupiter pair. I have found that the truncation error of multistep methods on two-body systems grows in time like t^2, and the roundoff like $t^{1.5}$, and I have a theory that accounts for

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Block 20 cont.

this. I have also tried to design better multistep integrators than the traditional Stormer and Cowell methods, and I have found a few interesting ones. A second result of my search for new methods is that I did not find any predictor that is stable on the Sun-Jupiter system, for stepsizes smaller than 108 steps per cycle, whose order of accuracy is greater than 12. For example, Stormer-13 becomes unstable at 108 steps per cycle. This limitation between stability and accuracy seems to be a general property of multistep methods.

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**

Scanned

Date: 8/23/1995

M.I.T. Libraries
Document Services