N-Body Simulations: The Performance Of Some Integrators: A Literature Review

Divye Gupta
B.Sc. Student
Saint Mary's University

1. Introduction

What is an N-Body simulation? How is a standard N-Body Simulation used to test initial-value simulators? Hold on! What are initial-value simulators?!

The article lacks the explanation of these and lot more terms before starting the core introduction. An N-body simulation is an approximation of motion of dynamic system of particles under the influence of some physical forces such as Newtonian Gravitational Force.

Dynamic system of particles can range from large celestial bodies in universe to individual atoms of a gas cloud. Here, we have four classic N-body problems that include only the Sun, Jupiter, Saturn, Uranus, and Neptune interacting through Newtonian Gravitational Force. Now you will think why we are just including Jovian planets for testing but not our terrestrial planets, that's because these bodies drive most of the dynamics of our Solar System.

The author leads the introduction to a brief contrast of simulation done with the standard problem and simulations done nowadays. He discusses four N-body problems in the article, namely: Jovian Problem, Nine Planets Problem, Spin Axis Problem and DE102 Problem and compares them to a number of integrators like DIVA, STEP, RKSUITE and many more. Wait, did I also forget to tell you about integrators? In simple words, it's a system that integrates and outputs the value given on its input. A software that approximates a solution of Initial Value Ordinary Differential Equations (IVODE). Here, Author mentions about two types of integrators: Symplectic Integrator and Non-Symplectic Integrator. A Symplectic Integrator is an initial value integrator that, in addition to approximating the solution of an IVODE, also conserves (i.e., keeps constant) particular properties of a given problem such as the energy, momentum, or mass whereas Non-Symplectic Integrator is an initial value integrator that does not have anything special built into the method to preserve specific properties.

As the title of the article suggests, our focus is on the performance of different integrators. Accuracy, efficiency and error growth are the main components that the author is concerned about. Non-symplectic integrators are used to compare these components.

In the next paragraph, the author runs with some extraordinary astrophysics terms, explaining the *power law* for position and energy. There are two states of when determining the error that comprises our power law. The first is when error is governed by truncation error or systematic round-off error, the error grows like t^2 where t is the time. A systematic round-off error is occurred when an initial value integrator is computing a the computations that look like *yi+1 = yi = h(other terms.* Systematic round off errors are not random and

therefore can never cancel out. Hence, they have the same sign on most of the steps. They affect the accuracy i.e. the velocity error grows like t^2 but not the precision (the energy grows like t) of a component. In the second state of power law, the error is governed by stochastic round-off error; the error grows like $t^{3/2}$ where t is the time. This error is completely random and the energy grows slower as $t^{1/2}$ but position or velocity error grows slower as $t^{3/2}$.

By analysing how the error grows we can tell what sort of effect is dominating the error growth. If the error grows like t^2 then truncation error is the dominating effect but if the error grows like $t^{3/2}$ then round-off error is the dominating effect.

Nonsymplectic Methods	*Trunc. Error*	*System. Error*	*Stoch. Error*
Position Error	t^2	t^2	$t^{3/2}$
Energy Error	t^1	t^1	$t^{1/2}$

Symplectic Methods	*Trunc. Error*	*System. Error*	*Stoch. Error*
Position Error	t^1	$t^2(?)$	$t^{3/2}$
Energy Error	$t^0 = 1$	t^1	$t^{1/2}$

Table 1: Showing behaviour of errors in Symplectic and Non-Symplectic methods.

The Author has concluded the introduction by giving a brief outline of other sections of the article. However, in this review we will only focus on the Jovian Problem and some of the integrators.

2. The Problems

Dr Philip, very straightforwardly begins by setting the base of the problems and giving information about the units of distance, time and mass. He talks about the position error, i.e. the difference between the positions predicted by the numerical solution and the "correct" positions as given by the reference solution. A reference solution is a high accuracy solution computed at great computational cost that is expected to be much more accurate than the standard numerical solution.

Explicit Runge-Kutta (ERK) and Runge-Kutta-Nyström (ERKN) pairs are suitable pairs for performing the integration. They provide an efficient way of finding accurate numerical solutions to initial value problems. Hence, our reference solution was calculated by an integration in quadruple precision using either 7-8 of ERK or the 10-12 of ERKN pairs.
The size of the error in the numerical solution is measured in the 2-norm i.e. L_2 norm *(Magnitude is the common term for norm)*.

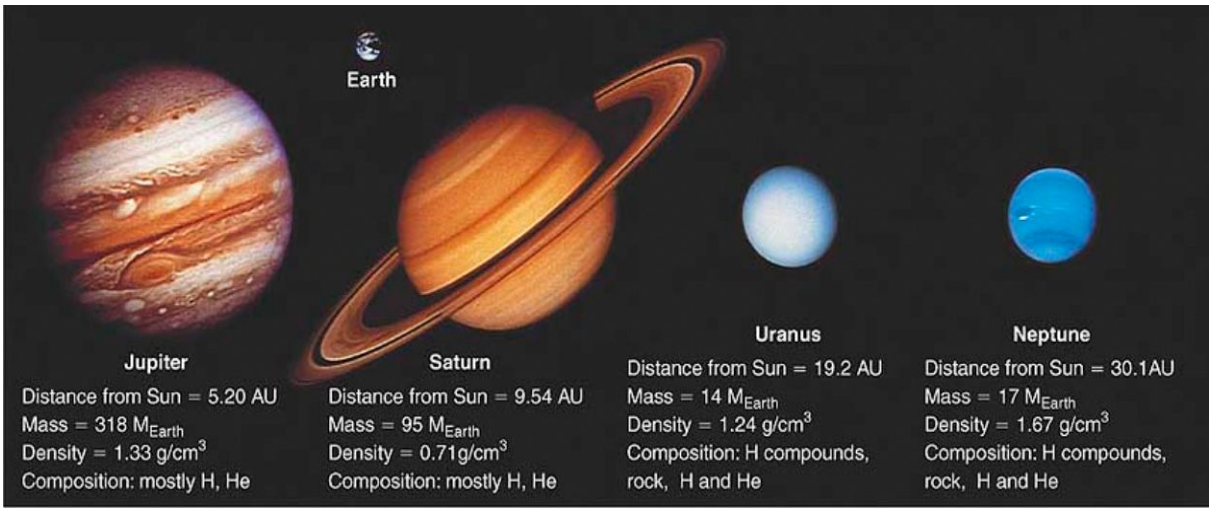


Figure 1: Showing the size of Jovian Planets compared to Earth.

2.1 Jovian Problem

As the name suggests, Jovian Problem consists of all the Jovian planets *(Jupiter, Saturn, Uranus, and Neptune)* and the Sun interacting through Newtonian Gravitational forces. The equations of motion of the Jovian problem are:-

$$\ddot{\mathbf{r}}_i(t) = \sum_{j=1, j\neq i}^{5} \frac{\mu_j(\mathbf{r}_j(t) - \mathbf{r}_i(t))}{\|\mathbf{r}_j(t) - \mathbf{r}_i(t)\|_2^3}, \quad i = 1, \ldots, 5,$$

Equation 1: The equations of motion of the Jovian Problem.

No, it's not as complicated as it seems. Here we are talking about two bodies *i* and *j.* The $\mathbf{r}_i$ is the position of the *i*-th body from the Sun. The position of the body is measured in terms of three dimensional standard Cartesian coordinates, (x_i(a), y_i(b), z_i(c)). The origin of these coordinates is the barycentre of the bodies i.e. the origin at the center of mass of the two bodies *(the Sun and the Jovian planet).* The dot operator on the left hand side of the equation denotes differentiation with respect to t. μ_j is equal to Gm_j where G is the gravitational constant (6.673×10−11 N) and m_j is mass of j-the j-th body given in Table 2. In the denominator of the equation, modulus with a base 2 is L_2 norm. The interval for these problems is ten million years which is literally huge! Even the Jupiter can orbit 843,418 times around the sun. Jovian Problem satisfies the Law of Conservation of Energy. Hence, total amount of energy is always conserved over a long period of time.

	x	y	z
Sun	0.9209498686328694E−03	0.2304166030756204E−02	0.9127217048523883E−03
Jupiter	0.3350285173564643E+01	−0.3471457282981824E+01	−0.1571236964688948E+01
Saturn	−0.8971584118413477E+01	0.2281986174163616E+01	0.1331251331416312E+01
Uranus	−0.1002083045458687E+01	0.1732581263930256E+02	0.7605737768120762E+01
Neptune	−0.2919365978874257E+02	−0.7716981025967714E+01	−0.2426332656583918E+01
Sun	−0.4665246664531984E−05	−0.3149154564335707E−05	−0.1269852543206254E−05
Jupiter	0.5580977902917778E−02	0.4959111982658174E−02	0.1991007074196164E−02
Saturn	−0.1862917203661242E−02	−0.4987007735981776E−02	−0.1981527265350456E−02
Uranus	−0.3959919409682252E−02	−0.3790629396772767E−03	−0.1101198438310003E−03
Neptune	0.8160828111535530E−03	−0.2775247414144566E−02	−0.1157385882979126E−02

Table 2: Initial conditions and parametric values for the Jovian Problem.

3. Integrators

When comparing the integrators we need to first know the properties of an integrator. Here we will first discuss the type of an integrator and then move on to the integrators included in that category.

- **Adams Integrator:** Adams Integrator is a type of initial value integrator that uses multi-step method. That is if you want to approximate a numerical solution at point t_1 then it depends on the numerical solution computed already at point t_0. Similarly, the numerical solution at a point k will depend on t_i as well as all of the previous points t_{i-1}, t_{i-2}, ...t_{i-k}.

  The first Adams Integrator discussed in the article, DIVA is a variable-order variable-stepsize integrator (VOVS). An integrator is said to be variable-order when it has a set of methods built into it that are of different orders. Order of a differential equation is the highest degree of derivatives in the equation. DIVA is used for q-th and mixed order problems. A system of equations is said to be mixed order if it contains derivatives of multiple orders. The Predictor, the rough approximation of the system, is of order k and the Corrector, refining the initial approximation with other means, is of order k + 1 where k <= 19.

  STEP is the second (VOVS) Adams Integrator discussed in the article. It's used for only first-order problems and predictor and corrector order for STEP are k and k + 1 respectively, where k <= 12. Note that the value of *k* can vary in DIVA for different equations but the value of *k* remains same in STEP. STEP uses compensated summation, that's an algorithm to reduce numerical error in the total obtained by adding a sequence of finite precision floating point numbers, compared to the obvious approach. For compensated summation we need to know about Local Error Tolerance (TOL), that is when you have a solution at point t and you want to take out the solution at point (t +

1). The estimate error at point (t + 1) is considered as TOL. We will discuss more about this term in Section 4. Compensated Summation is used to reduce the systematic-round-off error when TOL is less than 100 times the size of round-off associated with the floating point arithmetic.

- **Explicit Runge-Kutta (ERK) Integrator:** ERK integrator is a type of initial value integrator that uses a numerical method that computes an approximation of numerical solution at point t_{i+1} that depends on numerical solution it has already computed at t_i. It also depends on intermediate solution approximations it computes in between t_{i+1} and t_i. The computed solution, unlike Adams integrator, doesn't depend upon values previous to t_i.

  DXRK8 is a fixed-order variable-step (FOVS) integrator that advances the solution using an order-eight formula *(fixed order)* and step size is controlled by an embedded formula. 12 points are used to evaluate the derivative on each step.

  RKSUITE is another FOVS ERK integrator that uses suitable ERK pairs for integration. The pair uses 13 points to evaluate the derivative on each step.

  RKNINT is a FOVS ERKN integrator with a couple of ERKN pairs. The pair uses 17 evaluations per step. CS4 and CS7 are the only fixed step-size Symplectic integrators compared.
- **Stormer Methods:** The author next compares the performance of S9 and S13 integrators based on Stormer method of orders nine and thirteen respectively. Stormer Method is a multi-step method for second order equations, originally designed for solving Newton's equations of motion. It does so by converting the second order Ordinary Differential Equation (ODE) to first order system of equations and then estimating a solution.
- All the variable step-size integrators that we have discussed ensure that truncation error dominates over round-off error that is because the error grows like t^2. If the round-off error seems to be very large during an integration, it's recommended to go back to the previous step to save the time and process of the integrator.

4. Results

If the interval of integration is very long, it would definitely result in small error tolerances for variable-step integrators. Remember that all the variable step-size integrators ensure that truncation error dominates over round-off error so the value of TOL used for comparisons are 10^{-13} and 10^{-14}. The values below 10^{-13} and 10^{-14} may result in round-off error since round off error is directly proportional to *t* but truncation error is proportional ton t^2. So by using a value greater than 10^{-13}, we can also compare the performance of the integrators. We are not concerned about dominating the truncation error in fixed-stepsize integrators but we want to illustrate the effects of truncation error as well as round-off error. In the next section we will see the graphs of *E(t)* i.e. the relative position error with respect to position and velocity. Generally velocities of bodies are very large and that results in a very small value of *E(t)* but in our case, the bodies do not make close approaches and hence the velocities are not large.

Author briefly tells us how he measured computational effort i.e.: by using the number of derivative evaluations and CPU time but fails to tell the procedure that how it is actually measured. To estimate the dependency of CPU time and propagation of the round-off error comparisons were made on a 500 MHz SGI processor, a 700MHz IBM xSeries 350 processor, and a 2.2 GHz Dell PE2650 processor. Author states that all of the comparisons were in optimized, double precision FORTRAN but didn't mention that which sort of optimization was done.

4.1 Jovian Problem

Jovian Problem was solved using the order-13 Stormer method that had a stepsize of four long days. The large stepsize was used to make sure that the truncation error is small as well integrator is performing efficiently. A larger stepsize of 20 days was also used for S9 and S13 integrators and various other stepsizes like 10 and 100 days were used for CS7 and CS4. Here is the most important table for our integrations on the Jovian Problem:

	TOL	N_f	N_s	h	CPU	b	e_b	RMS	r
DIVA	10^{-13}	125,888,688	62,940,237	58.0	1186	2.094	0.0060	0.081	0.9960
DIVA	10^{-14}	131,723,154	65,855,590	55.5	1275	1.840	0.0026	0.036	0.9990
STEP	10^{-13}	235,292,227	117,171,519	31.2	1774	1.989	0.0015	0.020	0.9997
STEP	10^{-14}	303,568,678	151,350,267	24.1	2279	1.940	0.0034	0.046	0.9985
DXRK8	10^{-13}	873,329,016	72,777,417	50.2	2570	1.998	0.0011	0.014	0.9999
DXRK8	10^{-14}	1,063,794,551	88,649,544	41.2	3130	2.001	0.0011	0.014	0.9999
RKSUITE	10^{-13}	886,743,136	67,960,095	53.7	3883	2.003	0.0011	0.015	0.9998
RKSUITE	10^{-14}	1,219,896,644	91,726,459	39.8	5334	1.962	0.0019	0.025	0.9995
ODEX	10^{-14}	583,773,495	7,775,323	469.8	2003	2.007	0.0011	0.015	0.9998
ODEX2	10^{-13}	267,583,837	7,219,355	505.9	1126	1.994	0.0012	0.016	0.9998
ODEX2	10^{-14}	318,451,746	8,607,267	424.4	1338	2.014	0.0013	0.017	0.9998
RKNINT	10^{-13}	300,111,023	17,653,262	206.9	1074	1.944	0.0022	0.029	0.9994
RKNINT	10^{-14}	383,553,313	22,561,728	161.9	1373	1.869	0.0042	0.057	0.9975
CS4		146,100,001	36,525,000	100	475	0.635	0.0156	0.210	0.7904
CS4		1,461,000,001	365,250,000	10	4755	1.040	0.0050	0.068	0.9887
CS4		5,844,000,001	1,461,000,000	2.5	19023	1.129	0.0036	0.049	0.9950
CS7		438,300,001	36,525,000	100	1432	1.080	0.0100	0.135	0.9599
CS7		2,191,500,001	182,622,500	20	7158	2.025	0.0025	0.034	0.9992
CS7		4,383,000,001	365,250,000	10	14313	2.000	0.0013	0.018	0.9998
S9		182,625,154	182,625,000	20	2080	1.973	0.0013	0.017	0.9998
S9		913,125,154	913,125,000	4	10375	1.591	0.0052	0.070	0.9948
S13		182,625,210	182,625,000	20	2503	1.033	0.0013	0.017	0.9993
S13		913,125,210	913,125,000	4	12539	1.392	0.0055	0.075	0.9922

Table 3: Information on the Integrations for the Jovian Problem.

This table gives the cost of integration and the power law fit for the integrators. Power law fit means that we will try to fit the error vs. time data to a power law of the form error = t^b. This is done by taking the logs of the error and the time and considering a fit of the data to the line log ($error_i$) = b log(t_i) + c. That is we take the data from the experiments – this would be a collection of ordered pairs ($error_i$, t_i) – the error in the numerical solution at a given point in time – take the logs of this data, and then try to find the line that best fits

the data. This process is called Linear Regression or Least Squares fitting. This process would compute values for b and m that make the line as close as possible to the data.

The author does explain all of the terms very precisely and lights up the conclusions he made on the values observed. TOL is the Local Error Tolerance *(Recall that we mentioned that most of the comparisons were with TOL 10^{-13} and 10^{-14} since larger TOL leads to poor accuracy)*, N_f is the total number of derivative evaluations done in the integrator whereas N_s is the total number of "accepted" evaluations. Note that all of the steps that we take in solving a differential equations are not correct, sometimes our error become too large and it's worthless to continue our problem. Hence, we have to go back to our previous successful step and continue our problem with a different approach. N_s is the collection of those accepted steps. $\bar{h}$ is the average stepsize of an integrator and is calculated by summation of size of stepsizes taken for the integrator divided by the number of stepsizes. Standard error e_b in b is the difference between the value of b that we have computed from the power law fit to the actual value of b that is supposed to be in the theory of power law fit. RMS (Root Mean Square) is the common term that we learn in a statistics course. It is the least squares error associated with the Power Law fit i.e. Square root of the sum of the squares of the differences between the $error_i$ values and the value at t_i associated with the line obtained from the Power Law fit. Coefficient of correlation, r, is another statistics term that is a way of assessing how close a given set of data is to satisfying a linear relationship. The closer the correlation coefficient is to 1, the closer the data is to satisfying a linear relationship. We can clearly see from the table that no matter how big is our stepsize the valus of standard error (e_b) and the RMS is are really small, and the value of coefficient of correlation is very close to 1, hence the graphs shown below are accurately described by the power law.

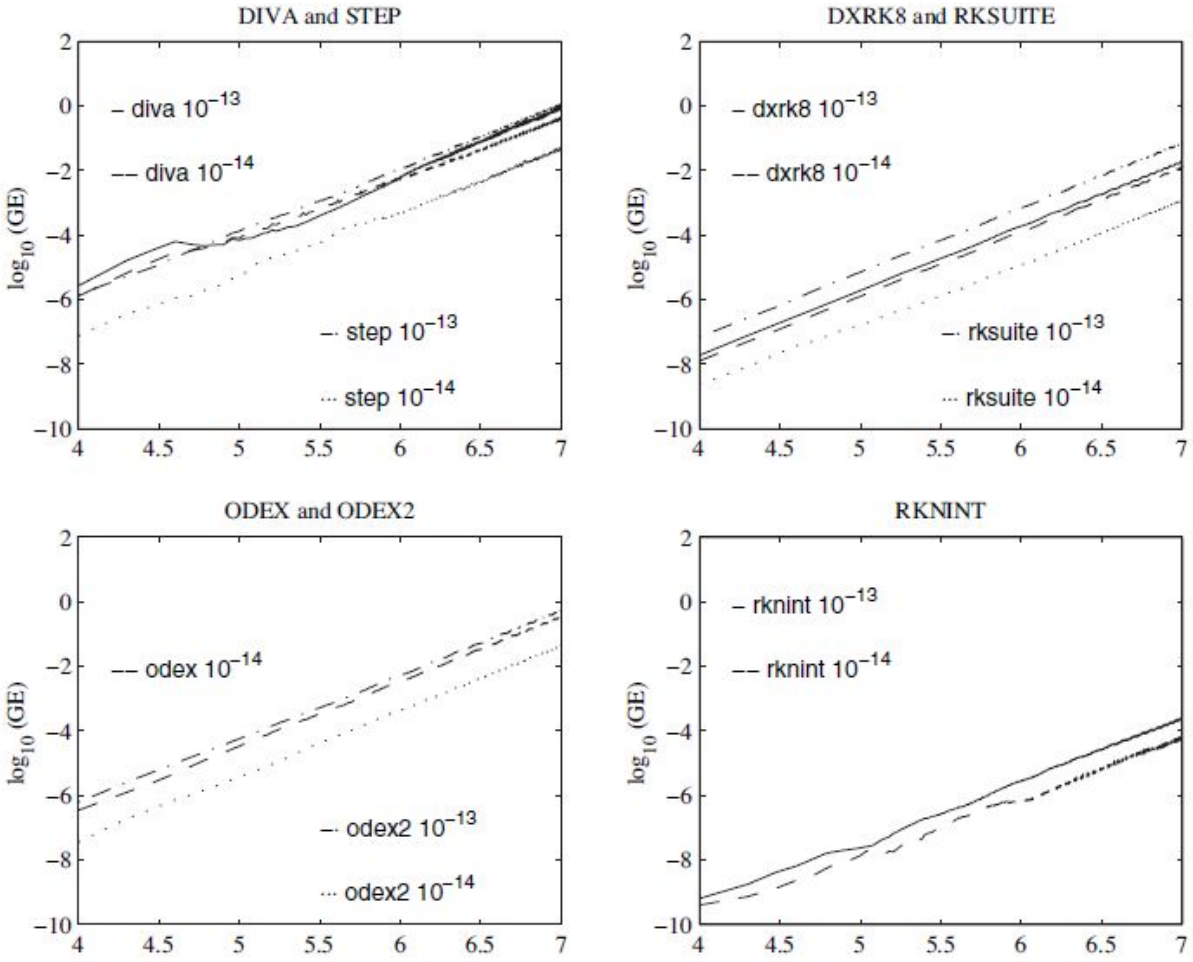
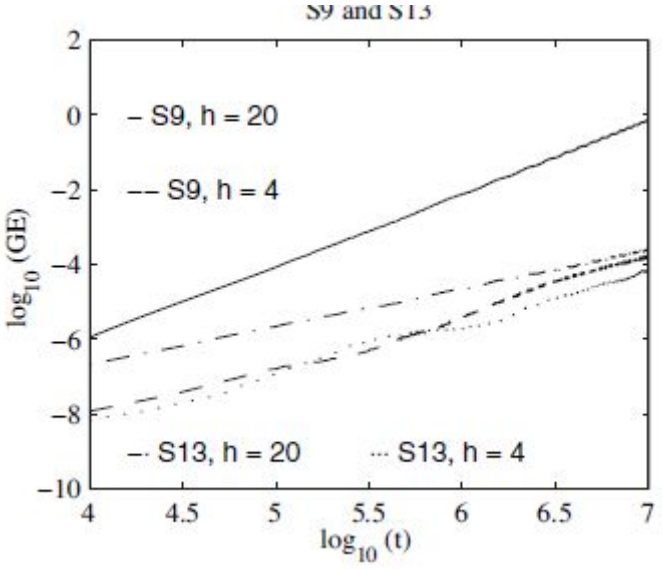
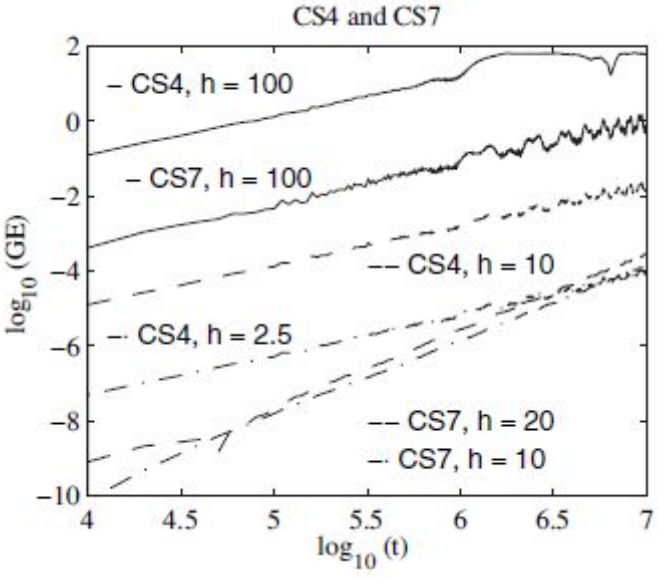


Figure 2.1: Showing Graphs of *E(t)*.

Now, look at the *E(t)* graph for the integrators STEP, ODEX2 and RKSUITE. The value of *E(t)* decreases as we go down from TOL 10^{-13} to 10^{-14} and shows that truncation error is dominated. Whereas if we see in RKNINT graph in Figure 2, the *E(t)* is dominated by round-off error. Also, the DIVA and DXRK8 show an increase in *E(t)* near the start of integrations with TOL = 10^{-14}, implying that the round-off error was dominant.



As per the theory, our results for S9 and S13 were quite accurate too. The value of *E(t)* with a stepsize of 20 assures that the round-off error is dominating but when you look at the value S13 value for stepsize 20, you will be muddled too. This mystery is left unsolved in the article by the author.



The author moves on to explain the results for CS7 and CS4 integrators. The best thing about the results sections is that author has beautifully compared the performance of every integrator and then stated the conclusion with those results. Going on to CS7 integrator for stepsizes 10 and 20, the b value is pretty close to 2 as compared to the values we got for a stepsize of 100 days. The *e(t)* increases expectedly for CS7: Stepsize 100 and CS4: Stepsize 2.5 and 10 but if you see the top curve for integrator CS4: Stepsize 100, the curve first increases for some years but then there is a slight drop leading to a slowly increasing curve. The author claims to prove a property of Symplectic methods that it indicates the oribital radius is known accurately but has not given much info for us to figure out that property.

Comparing the variable-stepsize integrators, you may have noticed that the graph of RKNINT starts from very bottom. This means that the value of *E(t)* was the smallest for RKNINT and CPU time taken by RKNINT is also less than anyone else *(expcept for DIVA with TOL = 10^{-14})*. Here, author fails to mention that ODEX2 also uses less CPU time than RKNINT for TOL = 10^{-14}. No matter how S9 and S13 start, the end point for *E(t)* is nearly 10^{-14}. RKNINT gave similar accuracy, but required less CPU time making RKNINT more efficient.

On comparing the Adams integrators for TOL = 10^{-13} we observed that after the first one million years the *E(t)* for both of the integrators was quite similar. But in the table, DIVA uses much less evaluations than STEP and less CPU time is taken. Hence, DIVA is clearly more efficient than STEP.

The author now compares ERK integrators DXRK8 and RKSUITE. Even though they used the same formulae to solve the problem, DXRK8 performed 48.8% more derivative evaluations than RKSUITE, which is a significantly large amount.

The author has used quadruple precision for the calculations to reduce the possible loss of significance that is an undesirable effect in calculations using floating-point arithmetic. It occurs when an operation on two numbers increases relative error substantially more than it increases absolute error, for example in subtracting two nearly equal numbers (known as catastrophic cancellation). Dr. Philip then summarizes the results by giving a brief comparison based on the graphs. He has included all the integrators in the short summary that simply states how is the relative error $E_E(t)$ is behaving in every graph *(We have already covered that in this section earlier).*

5. Discussion

Author has discussed all of the problems in this section and taken out the solution for the best integrator. Here, we will only discuss about our Jovian problem since that is the only one we have been working on throughout our review.

The best integrator throughout the problem has been RKNINT in author's point of view. As compared to the Adams Integrators: DXRK8 and RKSUITE, the RKNINT integrator can do 15 more derivative evaluations per accepted step. This is undoubtedly a great difference. This finding was concluded on working on the Nine Planets Problem.

When we bring the Stormer methods into the account the comparison becomes more complicated because the energy errors can grow according to different power laws in different problems. The CPU time taken by RKNINT is less than order-13 Stormer method for the intervals we used in our problem but when an estimate, beyond the original observation range was taken it suggested that order-13 Stormer method will eventually require less CPU time. This process of estimation is known as extrapolation.

Method of least squares generally to the approximate solution of overdetermined systems was used to fit the power law at^b to the norm of position error. The fit was good for most of the nonsymplectic integrators but there was one notable group exception in results for our Jovian Problem i.e. when stepsize was 100 for CS4 integrator. The value of b was off by approximately 1.4. The huge difference might be because of the large stepsize for this integrator. CS4 worked a little better with other stepsizes like 10 and 2.5 but still it has the

lowest b value as compared to all the other integrators.

In the concluding paragraph, the author states about the problems faced during rounding-off done on long simulations in double precision. The rounding-off could limit the achievable accuracy like we had in Jovian Problem, in which minimum error measured was in 10^{-4} after ten million years. If we need to have more detailed information, the power growth would increase much more than just ten million years and hence we should have the ability of higher precision arithmetic where very minute details would also be detected.