European Society of Computational Methods in Sciences and Engineering (ESCMSE)

Journal of Numerical Analysis, Industrial and Applied Mathematics (JNAIAM)
vol. 1, no. 1, 2006, pp. 59-80
ISSN 1790–8140

Variable Step Runge-Kutta-Nyström Methods for the Numerical Solution of Reversible Systems

J. R. Cash and S. Girdlestone
Department of Mathematics, Imperial College London
South Kensington London SW7 2AZ, ENGLAND

Received 2 February, 2006; accepted in revised form 10 April, 2006

Abstract: Fixed step, symmetric Runge-Kutta-Nyström formulae have proved to be very efficient for the numerical integration of a large class of reversible second order systems of ordinary differential equations of the special form $\frac{d^2y}{dt^2} = f(t, y)$. However for some important classes of problems it is necessary, for the sake of efficiency, to allow a variable steplength of integration to be used and in this case existing numerical methods tend to be much less satisfactory. In the present paper we examine in detail the problem of implementing reversible Runge-Kutta-Nyström integration formulae with varying time steps. We show that, even though enormous gains in efficiency can be made if the methods are implemented in an appropriate way, there are still some important practical problems to be overcome.

Keywords: Symmetric methods, variable step size, reversible system, Runge-Kutta-Nyström

Mathematics Subject Classification: 65L05, 65L06, 65L20

1 Introduction

In a recent paper [1] one of the present authors has developed a class of implicit Runge-Kutta-Nyström formulae for the numerical integration of ρ-reversible, second order ordinary differential equations of the form

$$\frac{d^2y}{dt^2} = f(t, y) \tag{1.1}$$

The aim of these numerical methods is to follow the long term behaviour of the solution of (1.1) rather than simply to obtain good local accuracy per se. It is usual to refer to such numerical methods as being geometric integrators. In particular a method can be regarded as being a geometric integrator if it possesses some appropriate stability properties as in the sense of [2] so that the dynamical properties of the original system of differential equations are inherited by the numerical approximation.

Perhaps the most famous example of geometric integration is associated with the numerical solution of Hamiltonian problems. Such differential equations take the form

$$\frac{dp}{dt} = -H_q(p, q), \quad \frac{dq}{dt} = H_p(p, q) \tag{1.2}$$

and the important property of Hamiltonian flow is that it is symplectic [3]. The aim now is to derive a class of numerical integrators which possess appropriate stability properties to allow the long term behaviour of the solution of (1.2) to be followed. When designing numerical integration methods we need to bear in mind that, since we normally wish to integrate over very long time intervals, it is the stability of the numerical integrator which is of prime importance. However, in simulations over finite intervals, the order of accuracy of the numerical method still has an important role to play.

It is fairly straightforward to derive good fixed step (symplectic) integrators for Hamiltonian problems (see [3] for example). However getting good variable stepsize methods is much more difficult and, in particular, Stoffer [4] has shown that symplectic integrators have essentially constant stepsizes (see also [11]).

Another important class of problems for which geometric integration methods are appropriate are ρ-reversible systems. These take the general form

$$\frac{dy}{dt} = f(y) \tag{1.3}$$

where the function f has the special property that $\rho f(y) = -f(\rho y)$ for all y where ρ is a linear map (see also [12]). A simple example of a ρ-reversible system is given by the partitioned ordinary differential equation

$$\frac{du}{dt} = f(u,v), \quad \frac{dv}{dt} = g(u,v) \tag{1.4}$$

where $f(u,-v) = -f(u,v)$, $g(u,-v) = g(u,v)$. In addition there are many important classes of problems that can be expressed in ρ-reversible form and for a discussion of this the reader is referred to [1, 3]. In particular it is well known that some Hamiltonian problems are ρ-reversible. For example, if the Hamiltonian function $H(p,q)$ satisfies $H(-p,q) = H(p,q)$, and this is the case when $H(p,q) = \frac{1}{2}p^T p + v(q)$ for example, then the Hamiltonian problem is reversible.

Of particular interest to us in this paper are special second order equations of the form (1.1). It is straightforward to show that such problems are ρ-reversible. It is interesting to note that if we denote the flow of our system by $\phi_t(y)$ it can be shown that for reversible systems

$$\rho\phi_t(y) = \phi_t^{-1}(\rho y) \tag{1.5}$$

and this accounts for the use of the word 'reversible'.

In [1] a new class of symmetric Runge-Kutta-Nyström methods were derived for the numerical integration of (1.1). It is quite natural to consider symmetric methods since they have been found to have excellent long term behaviour when applied to ρ-reversible systems since symmetric methods are themselves ρ-reversible [10]. Also, as we shall see later, symmetry plays a very important role in the integration of reversible systems. One of the main reasons which led us to derive a new class of Runge-Kutta-Nyström methods for the integration of ρ-reversible systems was due to the numerical results presented in [3]. These results show clearly that symmetric Runge-Kutta methods are surprisingly efficient for the integration of such problems. However the results given in [3] were for a fixed stepsize of integration and it is clear that there will be difficulties in extending this approach to variable stepsize. In addition the formulae presented in [3] did not have many of the important computational aspects that we have come to expect from modern numerical methods. In contrast, the Runge-Kutta-Nyström formulae developed in [1] do have some important properties that are not present in many other classes of integration formulae that have previously been proposed for ρ-reversible systems. In particular the methods of [1] have high stage order (which allows good predictors to be developed and continuous solutions to be defined), they have asymptotically correct local error estimates available (and this can be achieved because the local error of the class of Runge-Kutta methods considered is an exact differential),

they also have the alternative possibility of using embedding to obtain a local error estimate and, because of the ability to estimate the local error accurately, they have the possibility of using variable stepsize. Numerical results presented in [1] indicate that, when using a fixed steplength of integration, the results obtained by the new class of symmetric Runge-Kutta-Nyström formulae are very competitive with existing methods. However it was also clear from further experimentation that for some important classes of problems (such as Kepler's equations with eccentricity close to unity) it is very important, for the sake of efficiency, that the integration method should be able to use a variable stepsize. A preliminary investigation of the use of variable stepsize with a particular class of RKN formulae was carried out in [1] and it was shown that variable step can lead to a considerable increase in efficiency. It is now well known that classical step choosing algorithms are no longer appropriate when using variable step geometric integrators. In particular, Stoffer [4] has shown that when using variable step the step must be chosen so that the error estimate is satisfied exactly to machine precision. The crucial property of the Stoffer approach is that all computations must be carried out using symmetric formulae. In particular we need to construct our numerical algorithm so that the step from (p_0, q_0) to (p_1, q_1) is the same as would be taken from $(-p_1, q_1)$ to $(-p_0, q_0)$. This requires

a) The use of a symmetric integration formula (symmetric for constant stepsize)

b) A symmetric error estimator

c) A symmetric step choosing function

and all of these aspects are discussed in some detail by Stoffer [4].

The theoretical justification for using fixed step symmetric algorithms to integrate ρ-reversible systems has been given in [6]. It is shown that when a constant stepsize of integration is used, the numerical stability of the integration method can be explained by means of a backward error analysis. In particular, by using this approach, it can be shown that the numerical solution obtained by a geometric integrator with a fixed step length of integration is the exact solution of a perturbed differential equation which is itself ρ-reversible. In [5] this analysis was extended to the variable stepsize case. If an appropriate reversible step choosing algorithm is used, and by this we really mean the algorithm of Stoffer [4], then a backward error analysis for a one step method again shows that the numerical solution is the exact solution of a perturbed differential equation which is again ρ-reversible. This result explains some of the important geometric properties of the numerical flow such as the preservation of nearby invariants. In effect the analysis carried out in [5] extends what is known for constant stepsize to variable stepsize formulae when an appropriate step choosing algorithm is used.

We now consider the derivation of a symmetric step choosing algorithm. The traditional approach to accepting a solution after a single step of an integration formula has been carried out is first to impose a requested local error tolerance, Tol, and at each step to obtain an estimate E of the local error. If the integration is carried out from t_n to t_{n+1} the conventional approach is to accept the solution at t_{n+1} if

$$\|E\| \leq Tol \tag{1.6}$$

However, as is now well known, this strategy completely destroys the good properties of geometric integrators and this is manifested by the build up of errors. If we use a geometric integrator, with a suitable step choosing strategy, we obtain a linear growth of errors. However if we use a conventional (non-geometric) integrator we find that there is a quadratic build up of the error. Indeed for conventional integrators the error growth is quadratic no matter whether a fixed stepsize or a variable stepsize of integration is used. If we use a geometric integrator with an error criterion (1.6) then the desirable property of linear error growth is lost and the error again increases quadratically.

In his important paper Stoffer [4] showed that to preserve linear error growth with variable stepsize it is necessary to use the criterion

$$\|E\| = Tol \tag{1.7}$$

and h must be chosen so that this criterion is satisfied exactly to machine precision at each step.

It should now be clear, from what has been said previously in this section, that geometric integrators need very careful implementation. Indeed it is our opinion that there still remain some important issues which have not been fully resolved. One of the main difficulties is that geometric integrators are implemented very differently from the more familiar stiff (implicit) or non-stiff (explicit) integrators. In what follows we will briefly summarise some of these differences.

(1) For stiff equations it is usual to use implicit integration formulae which are solved using some form of Newton iteration. In contrast, when integrating non-stiff equations it is customary to use explicit methods. However for the geometric integrators proposed in this paper (and also used in [3]) we use implicit methods to enable us to get reversibility more easily but we solve the algebraic equations using direct iteration. In these circumstances it is not immediately obvious that it is worthwhile using a large steplength of integration. If we do use a large integration step then the direct iteration scheme may take longer to converge to machine precision so what we gain in the increase of the size of the integration step we lose in the extra function evaluations that are needed to obtain convergence. Conversely if a small steplength is used the direct iteration is likely to converge very quickly.

(2) The standard step choosing algorithm is ill conditioned [5, p265]. To overcome this, Hairer and Stoffer [5] suggest the use of a lattice approach where the steplength chosen is a multiple of 2^{-m} where m is a fixed integer.

(3) We do not have a large test set of challenging problems on which to test our numerical integrators. There is therefore the danger that our numerical methods will become tuned to the few available problems.

(4) In order to apply Stoffer's approach it is necessary to solve the non-linear algebraic equations to machine precision. Compensated summation can be used to enhance the accuracy but special problems still remain when trying to achieve such high accuracy.

2 The Derivation of Numerical Methods

In [1] we derived a class of ρ-reversible integration formulae for the numerical integration of the special class of second order equations (1.1). We derived these equations by focussing our attention on MIRK formulae and then using the special algebraic relationship that is satisfied by the coefficients of these formulae. Using this approach we were able to compute a formula pair for the computation of both y_{n+1} and y'_{n+1}. In this section we will consider a much more transparent approach to the derivation of our formulae. To do this we again consider the approach derived in [1] which was to apply a standard MIRK formula to both y and its first derivative:

$$y_{n+1} = y_n + \frac{h}{6}\left\{y'_{n+1} + 4y'_{n+1/2} + y'_n\right\} \tag{2.1}$$

$$y'_{n+1} = y'_n + \frac{h}{6}\left\{f(y_{n+1}) + 4f(y_{n+1/2}) + f(y_n)\right\} \tag{2.2}$$

Using the fact that the coefficients of MIRK formulae satisfy

$$\sum b_j a_{ji} = b_i(1 - c_i) \tag{2.3}$$

we derived the class of RKN formulae given by

$$y_{n+1} = y_n + hy'_n + h^2 \sum b_i(1-c_i)f_i \tag{2.4}$$

$$y'_{n+1} = y'_n + h \sum b_i f_i \quad . \tag{2.5}$$

The approach which we will adopt in the present paper is to consider symmetric RKN methods of the form

$$y_{n+1} = y_n + \frac{h}{2}\left\{y'_{n+1} + y'_n\right\} + h^2 \sum_{i=1}^{r} b_i f_i \tag{2.6}$$

$$y'_{n+1} = y'_n + h \sum_{i=1}^{s} \hat{b}_i f_i \tag{2.7}$$

where, in order to maintain symmetry, $b_i = b_{r+1-i}$ and $\hat{b}_i = \hat{b}_{s+1-i}$. If we now substitute from (2.7) into (2.6) for y'_{n+1} we obtain

$$y_{n+1} = y_n + hy'_n + h^2 \sum_i \left(\frac{\hat{b}_i}{2} + b_i\right) f_i \tag{2.8}$$

It is important to note that, in common with the formulae derived in [1], equation (2.8) is specially designed so that it has no term in y'_{n+1}. As a consequence we do not have to use (2.7) to obtain y'_{n+1} until we have obtained convergence for y_{n+1} from equation (2.8).

In what follows we will use the previous approach to derive a fourth order pair. From equations (2.6), (2.7) with $r = 2$, $s = 3$ we obtain

$$y_{n+1} = y_n + \frac{h}{2}\left(y'_{n+1} + y'_n\right) - \frac{h^2}{12}\left(y''_{n+1} - y''_n\right) \tag{2.9}$$

$$y'_{n+1} = y'_n + \frac{h}{6}\left(y''_{n+1} + 4y''_{n+1/2} + y''_n\right) \tag{2.10}$$

Eliminating y'_{n+1} equation (2.9) becomes

$$y_{n+1} = y_n + hy'_n + \frac{h^2}{6}\left(2y''_{n+1/2} + y''_n\right) \tag{2.11}$$

It now remans to specify a symmetric fourth order approximation to $y_{n+1/2}$ and this is given by

$$y_{n+1/2} = \frac{1}{2}\left(y_{n+1} + y_n\right) - \frac{5h}{32}\left(y'_{n+1} - y'_n\right) + \frac{h^2}{64}\left(y''_{n+1} + y''_n\right) \tag{2.12}$$

Eliminating y_{n+1} and y'_{n+1} from (2.12) we obtain

$$y_{n+1/2} = y_n + \frac{1}{2}hy'_n + h^2\left\{-\frac{1}{96}y''_{n+1} + \frac{1}{16}y''_{n+1/2} + \frac{7}{96}y''_n\right\} \tag{2.13}$$

Our final formula is now defined by (2.11), (2.13) and (2.10). This is exactly the same fourth order formula that was obtained in [1] but we feel that the approach we have just described is a much easier and more transparent way of deriving our formulae. It is now very straightforward to derive an embedded formula for the purpose of error estimation. We do this by applying the Trapezium Rule to both y_{n+1} and y'_{n+1}:

$$y_{n+1} = y_n + \frac{h}{2}\left\{y'_{n+1} + y'_n\right\} \tag{2.14a}$$

$$y'_{n+1} = y'_n + \frac{h}{2}\left\{y''_{n+1} + y''_n\right\} \tag{2.14b}$$

Using (2.14b) to eliminate y'_{n+1} from (2.14a) we have

$$y_{n+1} = y_n + hy'_n + \frac{h^2}{4}\left\{y''_{n+1} + y''_n\right\} \tag{2.14c}$$

In what follows we describe very briefly how to obtain a 6(4) formula and, from this, the general approach to deriving $p(p-2)$ pairs follows in a straight forward way.

A 6(4) embedded pair

The main formulae are

$$\begin{aligned} y_{n+1} &= y_n + \frac{h}{2}\left\{y'_{n+1} + y'_n\right\} + \frac{h^2}{24}\left\{y''_{n+1} - y''_n\right\} \\ &\quad + \frac{5}{12}\alpha h^2\left[y''_{n+1/2\,-\alpha} - y''_{n+1/2\,+\alpha}\right] \qquad (2.15a) \\ y'_{n+1} &= y'_n + \frac{h}{12}\left\{y''_{n+1} + 5y''_{n+1/2\,-\alpha} + 5y''_{n+1/2\,+\alpha} + y''_n\right\} \qquad (2.15b) \end{aligned}$$

Eliminating the term y'_{n+1} in (2.15a) by using (2.15b) we have

$$y_{n+1} = y_n + hy'_n + \frac{h^2}{12}\left\{y''_n + 5\left(1/2 + \alpha\right)y''_{n+1/2\,-\alpha} + 5\left(1/2 - \alpha\right)y''_{n+1/2\,+\alpha}\right\} \quad . \tag{2.15c}$$

To derive a formula for computing an approximation to y at the off step points $t_{n+1/2\,\pm\alpha}$ we put a sixth order interpolating polynomial through the data $\left(y_n, y'_n, y''_n, y_{n+1}, y'_{n+1}, y''_{n+1}\right)$. This polynomial is straightforward to compute and is given in [8] as

$$\begin{aligned} y_{n+\omega} &= A_{66}(\omega)y_{n+1} + A_{66}(1-\omega)y_n + h\left\{B_{66}(\omega)y'_{n+1} - B_{66}(1-\omega)y'_n\right\} \\ &\quad + h^2\left\{C_{66}(\omega)y''_{n+1} + C_{66}(1-\omega)y''_n\right\} \qquad (2.16) \end{aligned}$$

where

$$\begin{aligned} A_{66}(\omega) &= \omega^3\left(6\omega^2 - 15\omega + 10\right) \\ B_{66}(\omega) &= \omega^3\left(3\omega - 4\right)\left(1-\omega\right) \\ C_{66}(\omega) &= \frac{\omega^3}{2}\left(1-\omega\right)^2 \end{aligned}$$

Note that because of the relationships (2.15a), (2.15b) we are essentially computing a polynomial through $\left(y_n, y'_n, y''_n, y''_{n+1/2\,-\alpha}, y''_{n+1/2\,+\alpha}, y''_{n+1}\right)$.

If we now evaluate (2.16) at $t_{n+1/2\,\pm\alpha}$, where α is the Lobatto point $\frac{\sqrt{5}}{10}$, then these two equations together with (2.15a) give us three algebraic equations in the 3 unknowns, $y_{n+1}, y_{n+1/2\,+\alpha}, y_{n+1/2\,-\alpha}$. For the purpose of error estimation we use embedding based on (2.15a) and (2.9). The way in which these nonlinear algebraic equations are solved has to be considered with great care and we examine this in the next section.

3 Solution of the Nonlinear Algebraic Equations

One of the most important computational aspects of a variable step code concerns the efficient solution of the nonlinear algebraic equations defined by the numerical method. When stepping from t_n to t_{n+1} $(= t_n + h)$, not only do we need to determine values for y_{n+1} and y'_{n+1}, we also need to

determine the value of the stepsize h which will give the required solution with an estimated local error exactly equal to the requested tolerance. The numerical algorithms that will be discussed in this section are symmetric RKN formulae and, as was explained earlier in this paper, it is crucial that when our numerical algorithms are implemented everything is kept symmetric. In the previous section we derived fourth order formulae which are given by (2.11), (2.13), (2.10) and a local error estimate which is based on (2.14c). We also have available a fourth order continuous interpolant which is given by

$$y_{n+\beta h} = y_n + \beta h y'_n + h^2 \left[\left(\frac{\beta^2}{2} - \frac{\beta^3}{2} + \frac{\beta^4}{6} \right) y'_n + \left(\frac{2\beta^3}{3} - \frac{\beta^4}{2} \right) y''_{n+1/2} + \left(\frac{-\beta^3}{6} + \frac{\beta^4}{6} \right) y''_{n+1} \right] . \tag{3.1}$$

This interpolant has the important property that if we put $\beta = 1$ we again obtain the integration formula (2.11). Our aim now is to define an iteration scheme which allows us to compute y_{n+1} and y'_{n+1} from (2.11), (2.13), (2.10) with an error estimate, E, which exactly satisfies (to machine precision) the condition

$$\|E\| = Tol \tag{3.2}$$

where Tol is the user requested local tolerance. It is clear that we are essentially finding a solution for y_{n+1}, $y_{n+1/2}$ and h simultaneously and it is worth noting that the equation for h, which is derived from (3.2), is a scalar one. The natural way in which we might proceed to solve these equations is as follows:

Algorithm 1

(1) Guess h and then use the interpolating polynomial (3.1) to compute initial approximations $y^{(0)}_{n+1/2}$ and $y^{(0)}_{n+1}$ to $y_{n+1/2}$, y_{n+1} respectively.

(2) Compute approximations to $y_{n+1/2}$ and y_{n+1} by computing one iteration of (2.11) and (2.13).

(3) Compute an estimate of the local error in y_{n+1} by using (2.11), (2.14c).

(4) Compute a value of $h = h_{new}$ from (3.2)

If the error estimate satisfies (3.2) to machine precision then we accept our latest approximation to y_{n+1}, we compute y'_{n+1} from (2.10) and we then advance to the next integration step. If (3.2) is not satisfied we compute a new value of h from (3.2) and go back to step (1) of algorithm 1. Using this approach in which all approximations are computed simultaneously we have to be a little careful in computing the final value of y_{n+1}. This is explained in detail in [5, 7]. Basically the idea is that after having computed y_{n+1} with a steplength $h = h_{old}$ we compute a new value of the steplength h_{new} from (3.2). This is given by

$$h_{new} = [Tol/Err(h_{old}, y_n)]^{\frac{1}{p}} \tag{3.3}$$

where $Err(h_{old}, y_n)$ is the latest estimate of the local error and p is the order of the integration method. We note that the convergence criterion is based on approximations to y_{n+1} and the iteration stops essentially when values of $y^{(q-1)}_{n+1}$, $y^{(q)}_{n+1}$ and h_{old} are obtained so that

$$\|y^{(q)}_{n+1} - y^{(q-1)}_{n+1}\| < \varepsilon \tag{3.4a}$$

$$\|E - Tol\| < \varepsilon \tag{3.4b}$$

where ε is the machine precision. The important point we are making is that when we obtain convergence we do so with a step $h = h_{old}$. However we can compute the value of h_{new} from (3.3) without any extra function evaluations. What we now propose to do is to compute those values of y_{n+1} and $y_{n+1/2}$ that would have been obtained if we had performed one more iteration with step h_{new}. To achieve this without using any additional function evaluations we make use of our interpolating polynomial (3.1). We compute $y_{n+h_{new}}$ and $y_{n+\frac{h_{new}}{2}}$ from (3.1) and these are our new finally accepted approximations.

Despite the algorithm described above being the most natural way to proceed, our numerical experiments have indicated clearly that this is not the most efficient way of proceeding especially if high accuracy is requested. The reason for this is that we are predicting values of h using non-converged values of y. We have found it to be much better to iterate y to convergence with fixed h and then change h as we now explain.

Algorithm 2

(1) Predict h and use the interpolating polynomial to compute predicted values $y^{(0)}_{n+1}$, $y^{(0)}_{n+1/2}$.

(2) Iterate the values of y to convergence (to machine precision) keeping h fixed.

(3) Once we have convergence compute an updated value of h from (3.3) and again iterate to convergence.

(4) Continue this until (3.4a), (3.4b) are satisfied.

This procedure is likely to be a very efficient one firstly because we have a good interpolating polynomial and so can predict $y^{(0)}_{n+1/2}$, $y^{(0)}_{n+1}$ very accurately. Second we have the advantage that we are predicting new values of h using converged values of y.

One possible problem with the two algorithms described earlier is that we may be getting convergence of our iteration schemes but it may not be sufficiently rapid. What we would like to do is to terminate our iteration scheme if convergence is slow or if the iterates are converging to a solution which will be unacceptable. We do this as follows

Algorithm 3

Guess h_{old} and compute values $y^{(0)}_{n+1/2}$, $y^{(0)}_{n+1}$. At each ith iteration we estimate $\|Tol - E\|_i$. If

$$\|Tol - E\|_i > \frac{1}{10}\|Tol - E\|_{i+1}$$

and this can only be computed if at least 2 iterations have been performed, then we terminate the iteration, compute a new value for h, compute interpolated values for $y^{(0)}_{n+1/2}$, $y^{(0)}_{n+1}$ and start again.

We will give the numerical results obtained for each of these algorithms in the next section.

4 Computation Aspects

In this section we examine some of the computational aspects associated with our integration formulae. The first thing we consider is the choice of the steplength of integration and to do this we follow the ideas of Stoffer [4] very closely. Stoffer suggests that, due to the nature of the step-size function, the iteration scheme which defines the step will converge rapidly if we use the

Secant method in the logarithmic plane. Suppose now that we start off our new computation by choosing an initial stepsize $h^{(0)}$. The local truncation error will then be of the form

$$err \approx \phi \left(h^{(0)}\right)^{p+1} \tag{4.1}$$

where p is the order of the embedded formula used to compute a local error estimate. Our aim is to choose a value h_{opt} which is such that

$$Tol = \phi \left(h_{opt}\right)^{p+1} \tag{4.2}$$

From (4.1) and (4.2) we have

$$h_{opt} \approx h^{(1)} = \left(\frac{Tol}{err_0}\right)^{\frac{1}{p+1}} \tag{4.3}$$

Once we have two approximations $h^{(0)}$, $h^{(1)}$ to h_{opt} we can perform our iterations using a secant method and this gives

$$h^{(i+1)} = \exp\left[\ln\left(h^{(i)}\right) + \left(\frac{\ln\left(\frac{h^{(i)}}{h^{(i-1)}}\right)}{\ln\left(\frac{err_i}{err_{i-1}}\right)}\right)\ln\left(\frac{tol}{err_i}\right)\right] \tag{4.4}$$

We can iterate this scheme to convergence to obtain h_{opt}. Assuming that we have obtained convergence to h_{opt} our next problem is to predict h for the next step. Clearly we should do this using some form of extrapolation. Assuming that at previous steps we have used h_{-1}, h_{-2} etc we could predict the next value of h using any of

$$\begin{aligned} h &= h_{-1} \\ h &= 2h_{-1} - h_{-2} \\ h &= 3h_{-1} + 3h_{-2} - h_{-3} \end{aligned}$$

or we could use higher order extrapolation. Sometimes of course when h is changing quite frequently a high order extrapolation formula would give poor results (even a negative prediction for h can be obtained on occasions) but to counter this we always have the fall back option of choosing $h = h_{-1}$ if our step prediction is unsatisfactory.

4.1 Explicit versus Implicit Integration Methods

In this section we compare the performance of an explicit and an implicit method on a ρ-reversible system. Numerical results presented in [1] and elsewhere show the great efficiency of variable step methods compared with fixed step methods on difficult problems such as those considered later in this section. What we wish to do at present is to compare the performance of the 4(2) method given by (2.10), (2.11), (2.13) with that of the 4(3) method given by Calvo and Sanz Serna [11]. The results obtained are shown in Figures 1, 2, 3, 4. We see that for all eccentricities tried the performances in fixed stepsize mode of the 4(3) and 4(2) formulae are fairly comparable. When we go on to consider a variable step implementation we see that for $e = 0.6$ and 0.9 the performances of the two codes are roughly comparable. However as the eccentricity approaches unity we see that the performance of the implicit method becomes much superior.

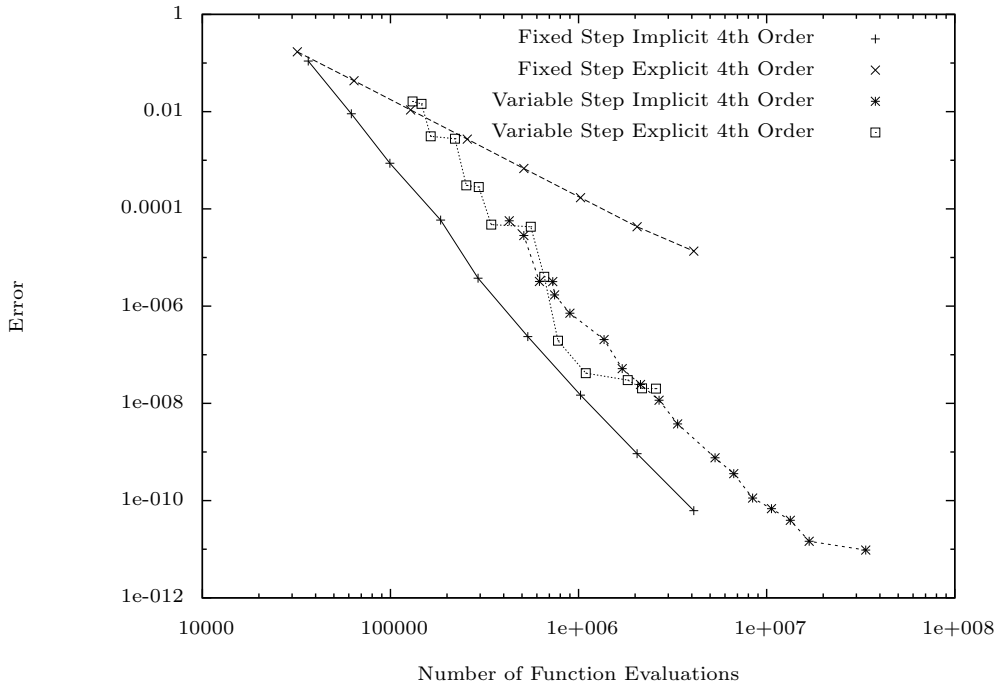

Figure 1: 4th order, $e = 0.6$, integration period 128π

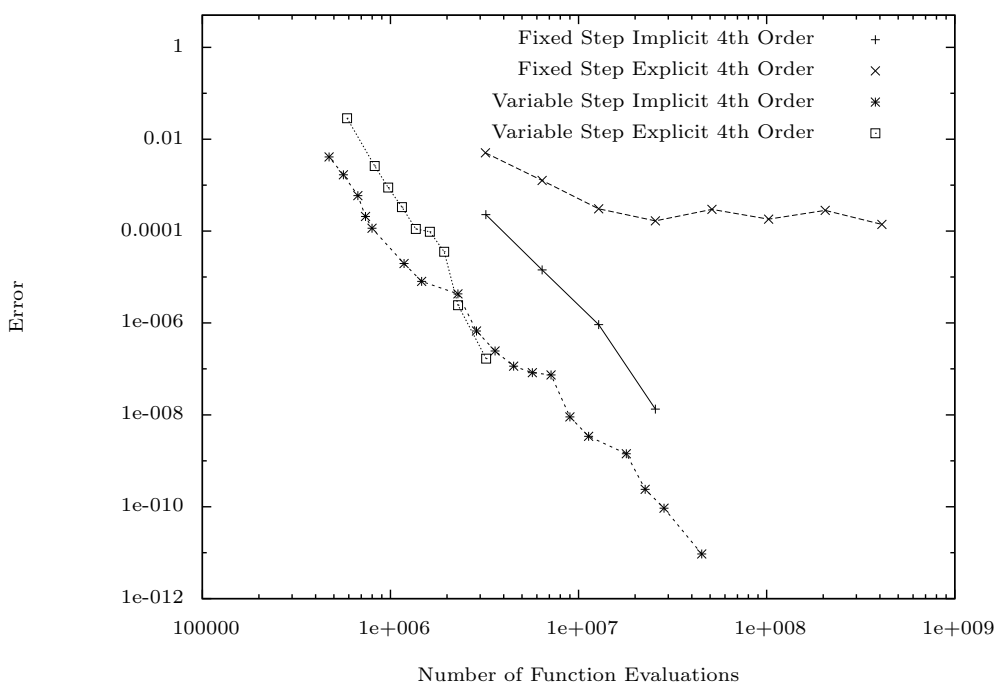

Figure 2: 4th order, $e = 0.9$, integration period 128π

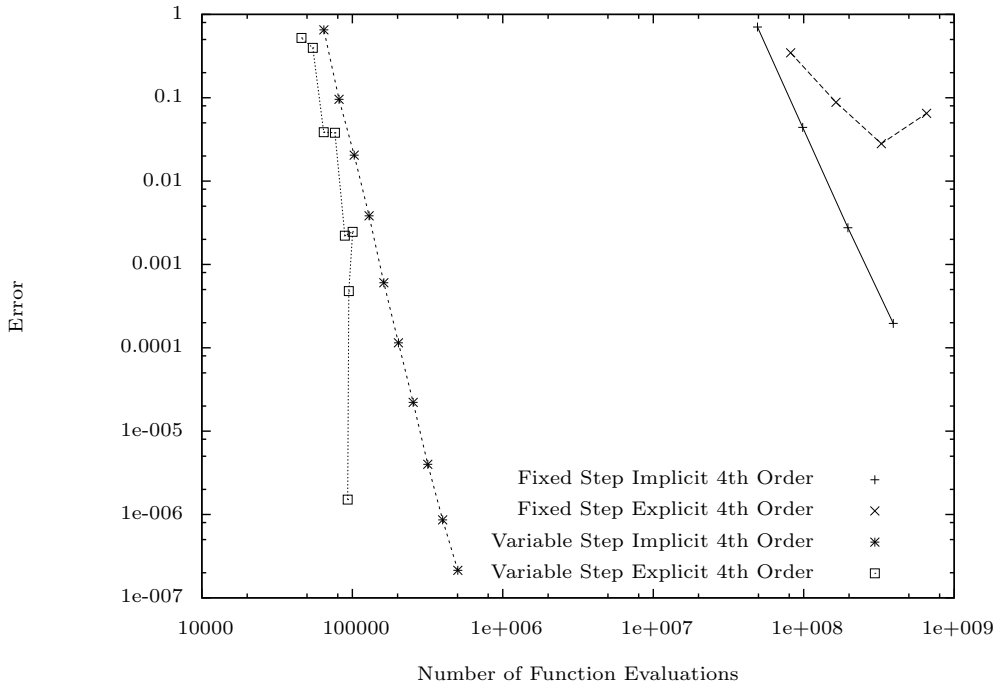

Figure 3: 4th order, $e = 0.999$, integration period 2π

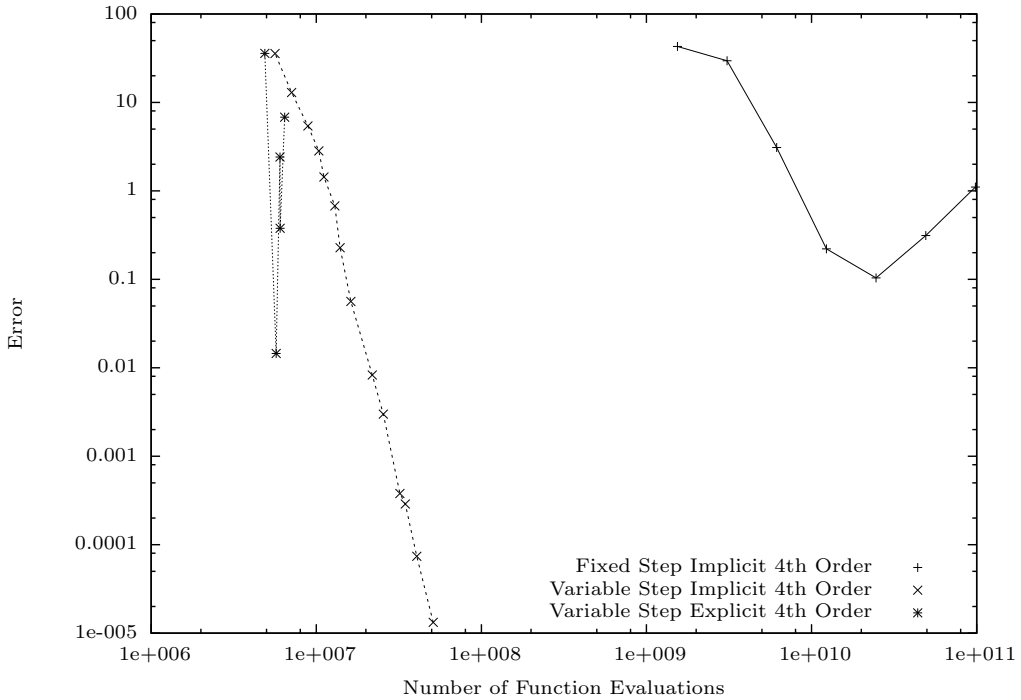

Figure 4: 4th order, $e = 0.999$, integration period 128π

4.2 Higher Order Methods

Previously in this chapter we have given the results obtained for a method of order 4 with an embedded method of order 2 being used for local error estimation. In this section we examine a method of order 6 which belongs to the same class of RKN formulae considered earlier. We will also derive an embedded fourth order method for the purpose of error estimation. All of the improvements that we developed for the fourth order method can immediately be applied to the sixth order method as well. However there is one major difference with the sixth order method and this concerns the local error estimate. The difference now is that we have the option of deriving a 6(4) formula or a 6(2) formula. When deriving a 6(4) formula in the standard way, i.e. using (2.15c) and (2.11) as an embedded pair, it is necessary to compute an additional function evaluation to derive the fourth order formula and the question is whether this extra function evaluation is worthwhile or whether it would be better to use the less expensive 6(2) formula. Our numerical experiments indicate that it is marginally better to use a 6(4) formula and that the performance of this formula becomes better as the problem gets harder (the eccentricity in Kepler's equations approaches unity).

4.3 Solving the Nonlinear Algebraic Equations

As was explained earlier in this paper there are at least three ways in which the nonlinear equations defining y_{n+1}, h and various intermediate values of y in range $[t_n, t_{n+1}]$ could be solved. The obvious way is to solve all of the equations simultaneously and this is what was done by Hairer and Stoffer. However our numerical experiments have shown that the most efficient approach is to use algorithm 2. Perhaps the reason for the effectiveness of this approach is that we are choosing h based on converged values of y. We give the numerical results obtained with the 3 algorithms described earlier with our 6(4) formula in Figures 5,6,7. We see that as the eccentricity approaches unity algorithm 2 becomes the most efficient.

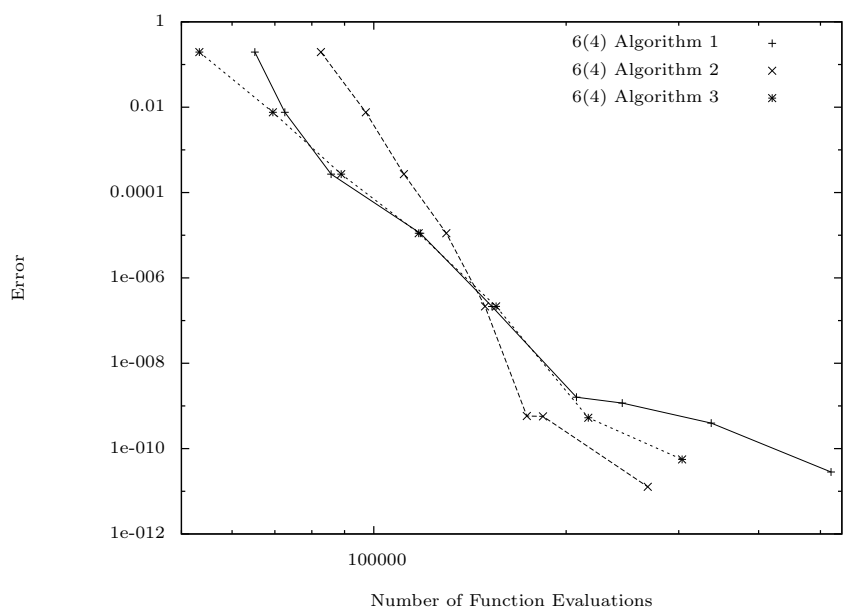

Figure 5: 3 Algorithms with 6(4), $e = 0.6$, integration period 128π

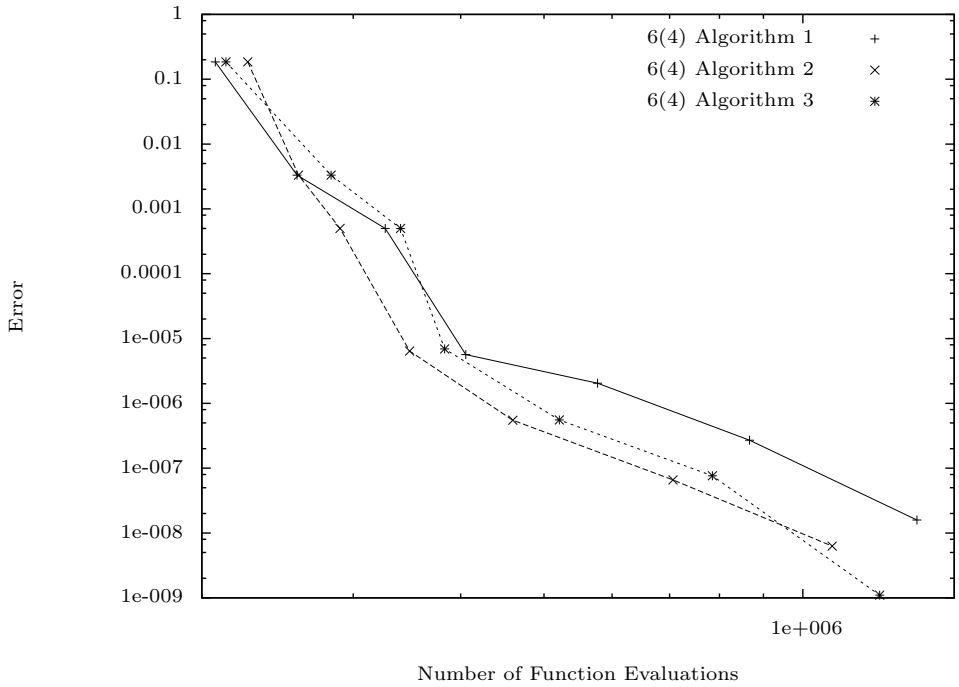

Figure 6: 3 Algorithms with 6(4), $e = 0.99$, integration period 128π

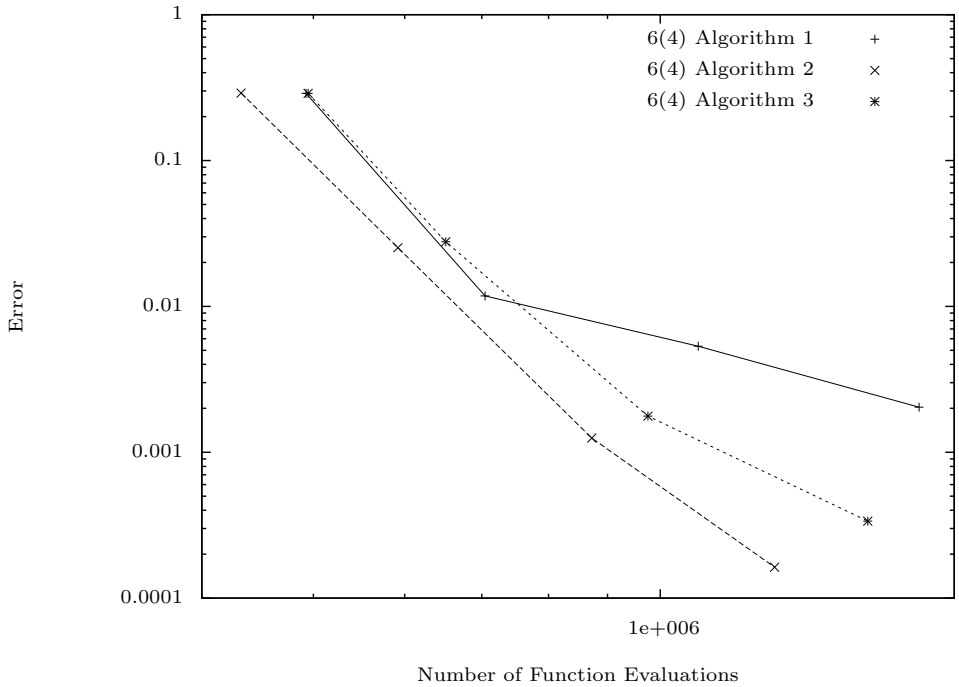

Figure 7: 3 Algorithms with 6(4), $e = 0.9999$, integration period 128π

4.4 Improving the Variable Step Selector

In this section we describe some changes we have made to the stepsize selector. In [4] it was suggested that we should compute h so that the local error is exactly equal to the requested tolerance at each step. The proposal we made in [1] was to allow the integration to continue with the steplength currently being used if

$$\frac{Tol}{10} < err < 10 \times Tol \tag{4.5}$$

Here $10 \times Tol$ is the user requested accuracy and err is the estimated local error. As we will see later, if we do allow this facility then we have to change our step choosing algorithm when a step is rejected. Of course we still have to impose the requirement that when the stepsize is changed it must be carried out so that $err = Tol$ to machine precision since this will ensure that the important symmetry ideas of Stoffer are maintained. Naturally the factor 10 appearing in (4.5) is not set in stone. We could replace 10 by a factor S and the larger the value of S is the more relaxed is condition (4.5) and the less often we need to change stepsize. Conversely, as the factor S approaches unity the more often we need to change the stepsize but the closer we are to Stoffer's original ideas. Our numerical experiments have shown that the smaller the value of S is the better the algorithm performs at very strict tolerances while at less stringent tolerances a larger value of S tends to perform better. This suggests that S might best be chosen as a function of the tolerance and we will return to this later.

If we are going to include a step control algorithm of the form (4.5) in our integration scheme we have to make sure that we continue to keep everything symmetric. This leads us to make a few changes in our step selection strategy as we now explain. Suppose our steplength algorithm is based on the slightly modified formula (4.5) which is

$$\frac{Tol}{S} < err < S \times Tol \tag{4.6}$$

We integrate forward until we reach a point where we have to change stepsize because (4.6) is violated. Perhaps the most obvious approach now would be to change the stepsize so that $err = Tol$. However to maintain symmetry we require that if (4.6) is violated because $err > S \times Tol$ then we choose h so that $err = Tol/S$ (exactly to machine precision). It is clear that if we now integrate backwards it is the lower limit condition that will be violated. In conclusion our algorithm is such that

(1) If $err > S \times Tol$ then we choose h so that $err = Tol/S$

(2) If $err < Tol/S$ then we choose h so that $err = S \times Tol$

(3) If (4.6) is satisfied then we continue to integrate with h held fixed.

To see how this new method performs we compare it to the original Stoffer method in Figures 8,9,10,11, 12. We will refer to the new approach as relaxed Stoffer. As can be seen, the new algorithm gives a considerable improvement over the old one.

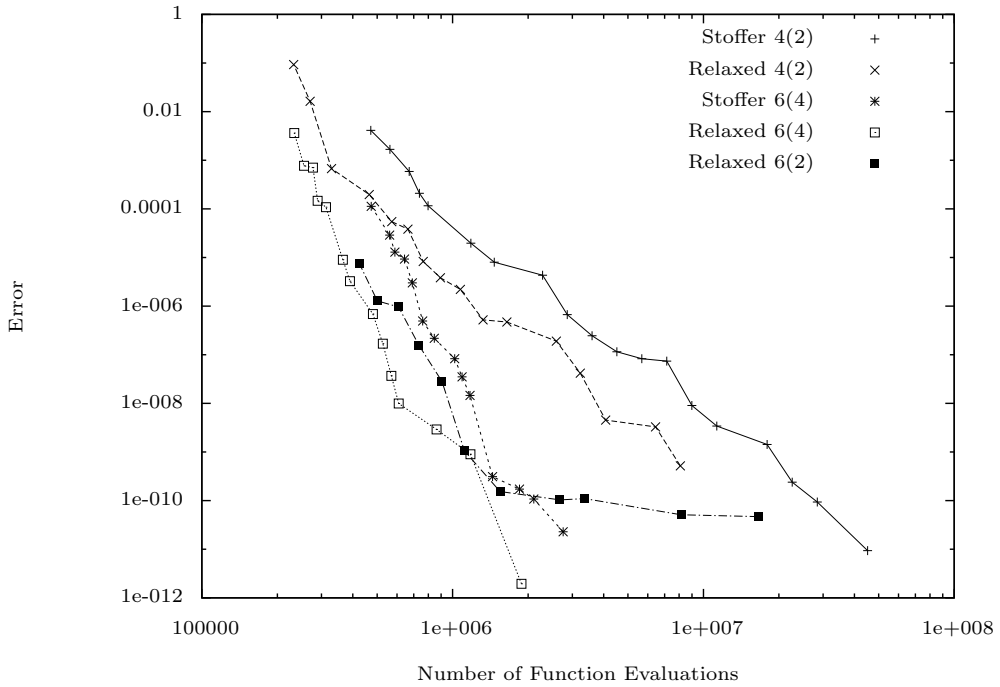

Figure 8: Stoffer vs. Relaxed method, $e = 0.6$, integration period 128π

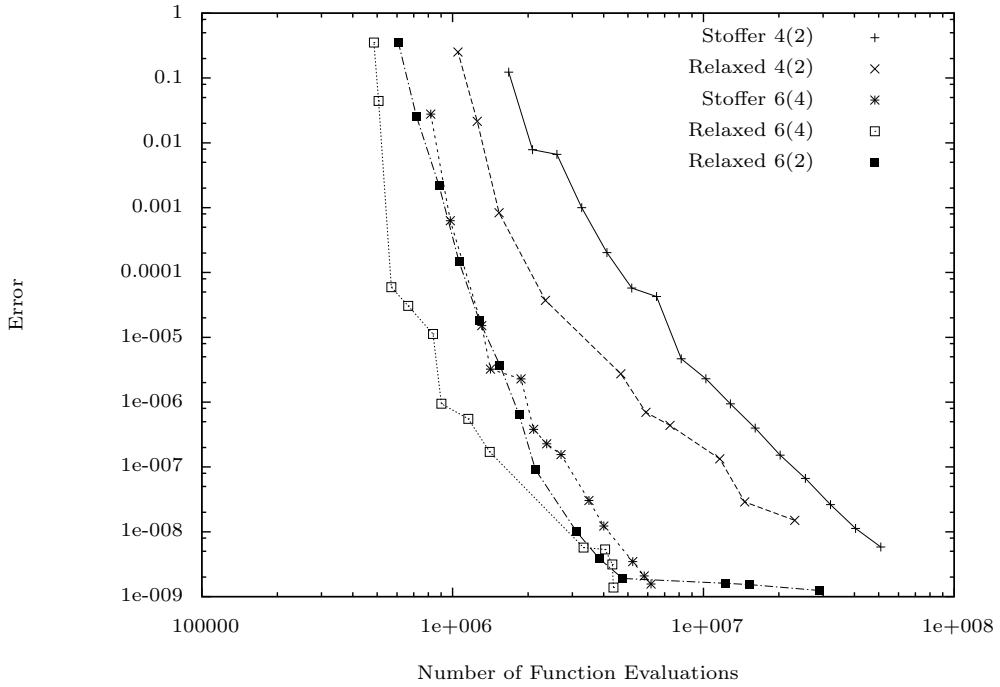

Figure 9: Stoffer vs. Relaxed method, $e = 0.99$, integration period 128π

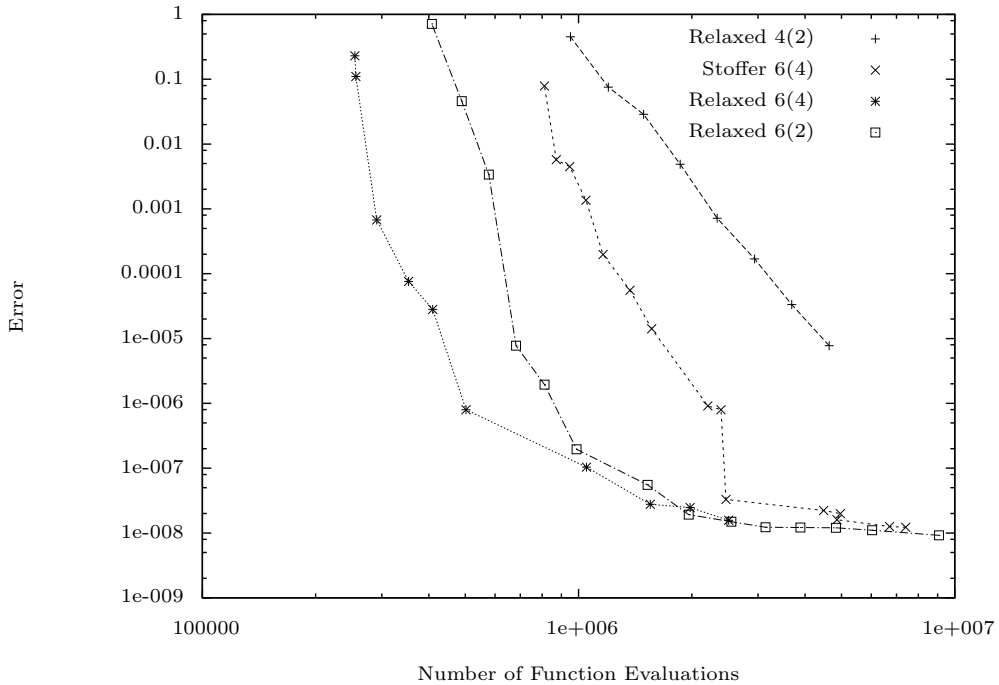

Figure 10: Stoffer vs. Relaxed method, $e = 0.999$, integration period 32π

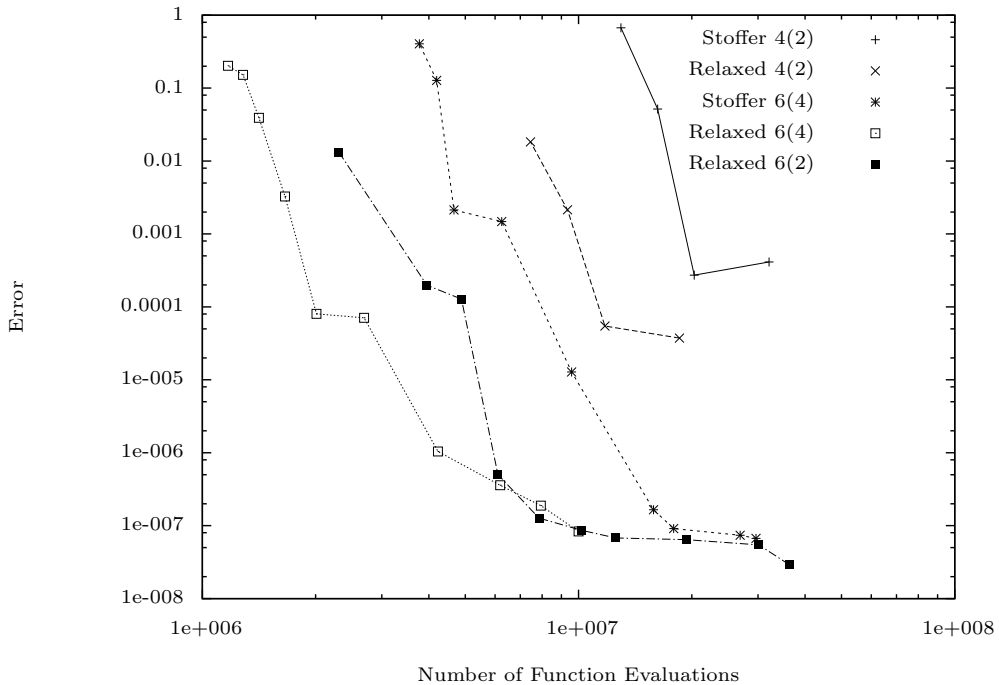

Figure 11: Stoffer vs. Relaxed method, $e = 0.999$, integration period 128π

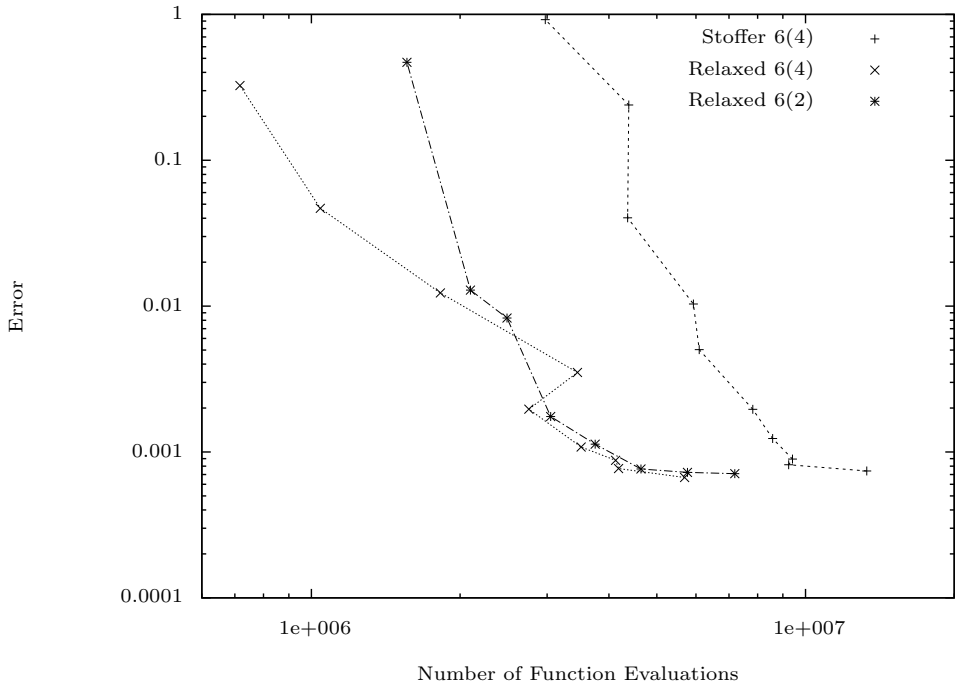

Figure 12: Stoffer vs. Relaxed method, $e = 0.9999$, integration period 32π

4.5 Arbitrary Precision

All of the numerical results contained in this paper are from running programs on a 32-bit computer using double precision (64-bit). If we implement the same programs (with a few minor modifications) on a 64-bit computer we can obtain 128-bit accuracy (about 32 decimal places). This would allow the solution of even harder problems as much smaller steps can be taken without rounding errors becoming significant. Also we would be able to run existing problems over much longer time intervals. Recently libraries such as ARPREC have been created for high level languages such as C and Fortran to allow very high precision to be obtained. In this section we carry out some experiments in high precision (in fact quadruple) arithmetic.

The first problem we consider is to see whether we can run harder problems which could not be solved using double precision arithmetic. We developed an eighth order formula with either an embedded formula of order 4 (denoted 8(4)) of order 6 (8(6)) and in figures 13, 14, 15, 16, 17 we give the results obtained on some much harder problems. The conclusions to be drawn from the figures is clear. In figures 13, 14 we see the much better performance of 8(6) compared with the other algorithms. The 6(4) and 6(2) formulae were not able to solve the problems in figures 15-16 and here we see the better performance of 8(6) in figure 15 while in figures 16, 17 the 8(6) formula is the only one that can solve the problem. Finally we present the results obtained for $e = 0.9951$. This is an important example since it is the eccentricity of the famous Hale-Bopp comet [1]. In figure 18 we plot the work precision diagram for the 8(4), 8(6) formulae in double and quadruple precision. We would expect that the double precision algorithms would be more efficient than the quadruple precision initially since the quadruple precision algorithms have to iterate the algebraic equations to machine precision. However for small errors the quadruple precision algorithms become much more

efficient as expected. Finally we consider running the problem much further. We note from figure 19 the excellent performance of the relaxed 8(6) algorithm when run on the Hale-Bopp problem in the range $[0, 20,000\pi]$. These results indicate that in spite of the extra run time required when quadruple precision is used the results obtained are often excellent.

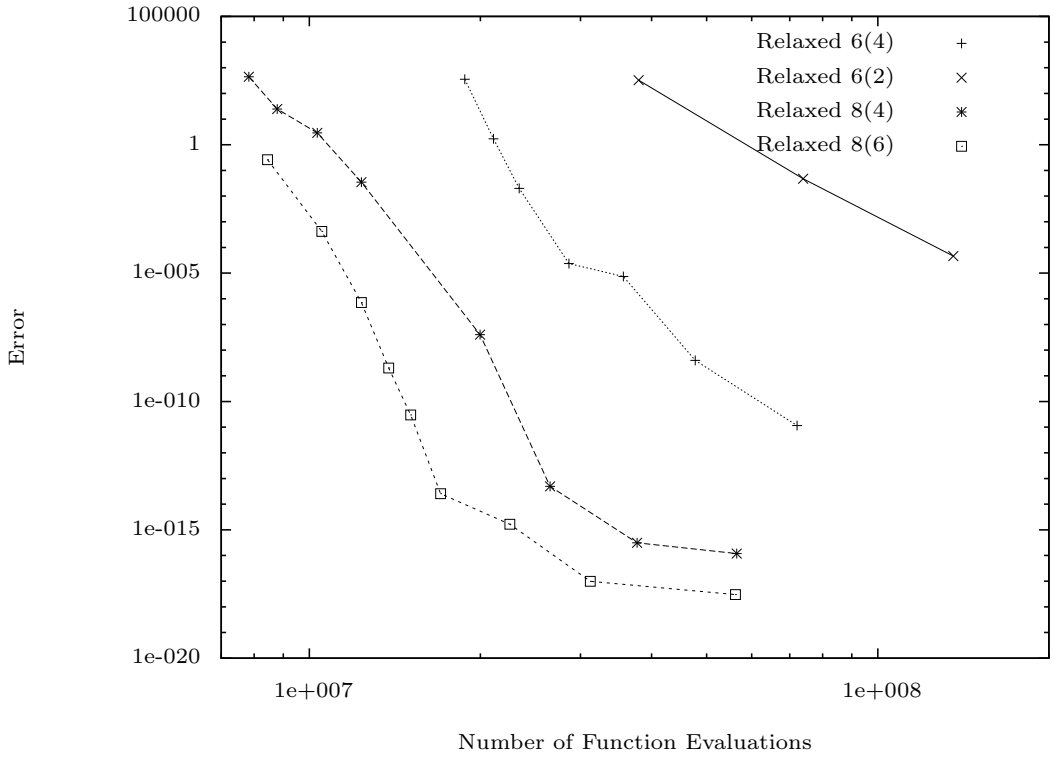

Figure 13: Quadruple Precision, $e = 0.99999$, integration period 128π

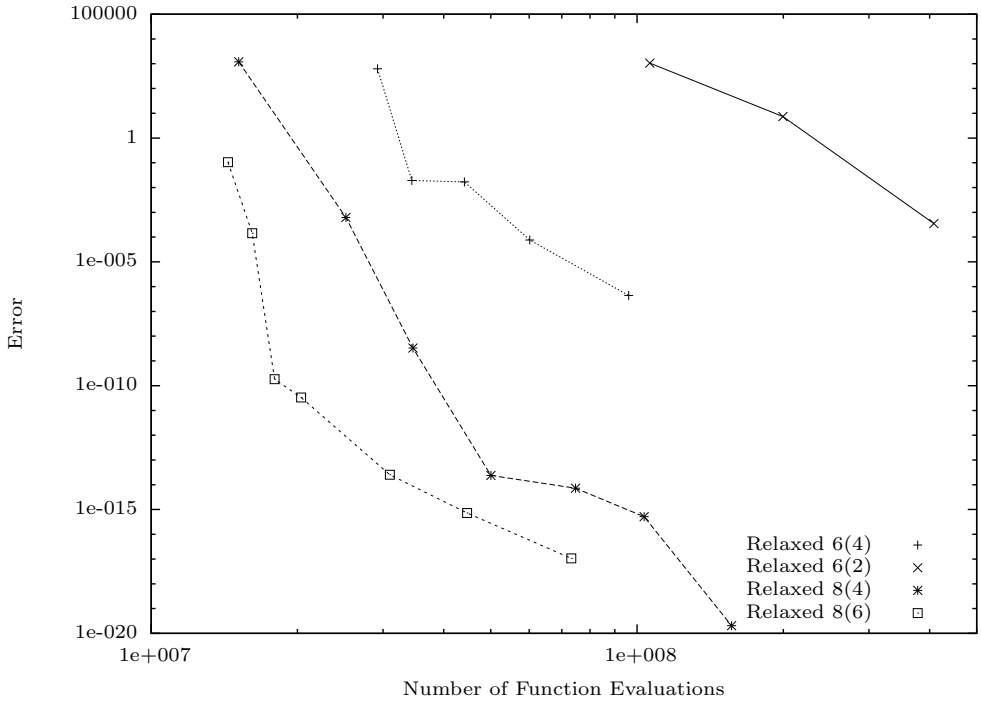

Figure 14: Quadruple Precision, $e = 0.999999$, integration period 128π

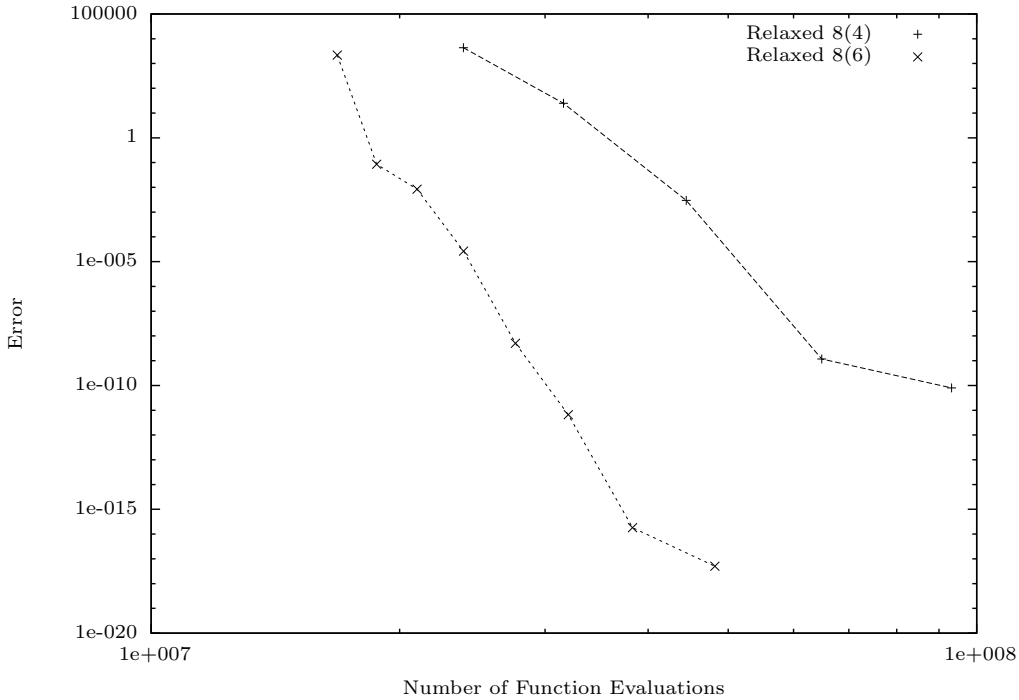

Figure 15: Quadruple Precision, $e = 0.9999999$, integration period 128π

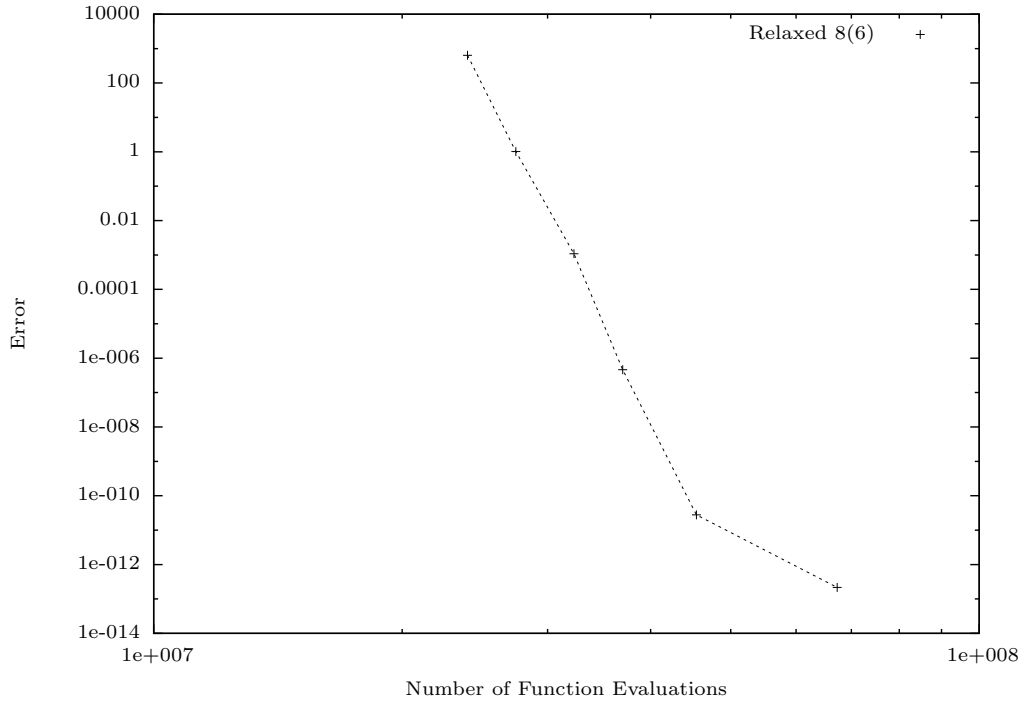

Figure 16: Quadruple Precision, $e = 0.99999999$, integration period 128π

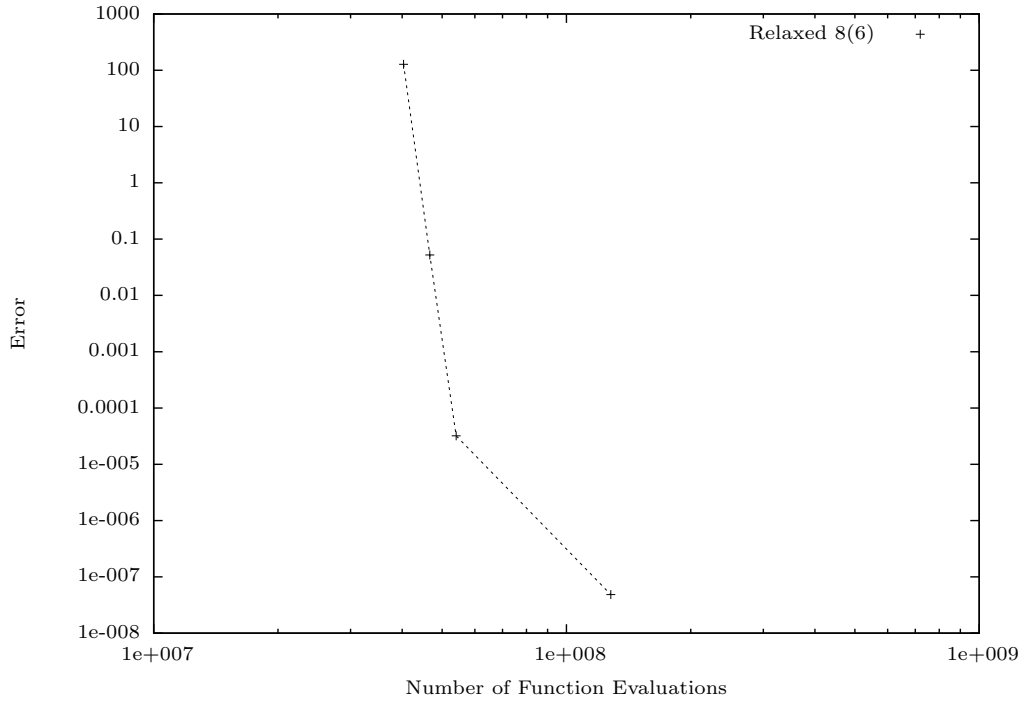

Figure 17: Quadruple Precision, $e = 0.999999999$, integration period 128π

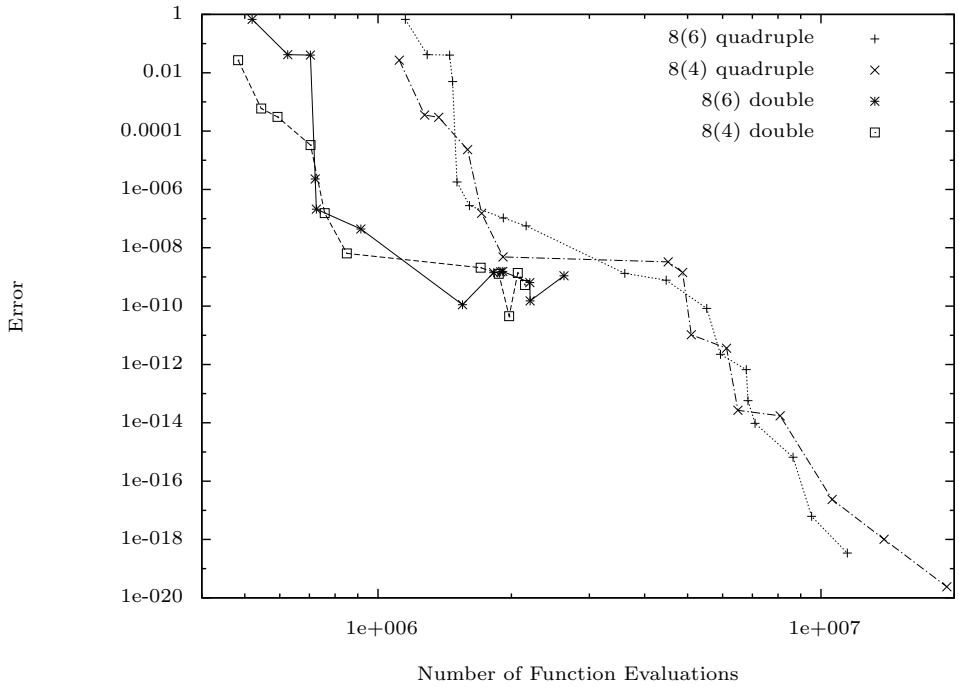

Figure 18: Quadruple vs. Double Precision, $e = 0.9951$, integration period 128π

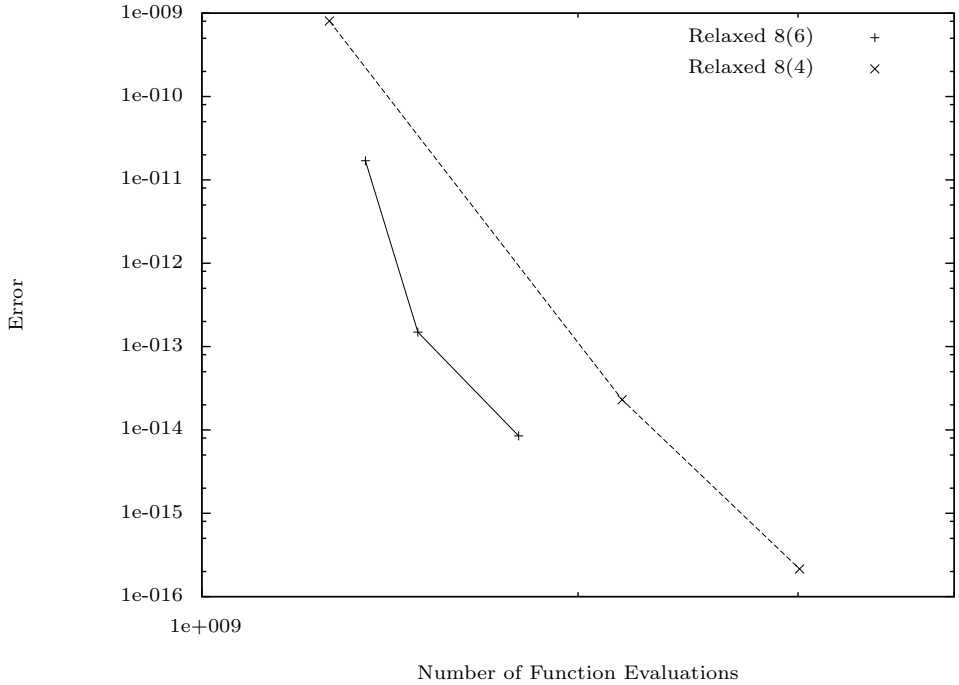

Figure 19: Long time integration of Hale-Bopp, $e = 0.9951$, integration period $20,000\pi$

References

[1] J.R.Cash, A variable step Runge-Kutta-Nyström integrator for reversible systems of second order initial value problems, SIAM J. Scientific Computing, 26, 963-978 (2005)

[2] A.M.Stuart and A.R.Humphries, Model Problems in Numerical Stability Theory for Initial Value Problems, SIAM Rev., 36 (1994) pp226-257

[3] E.Hairer, C.Lubich and G.Wanner, Geometric Numerical Integration, Springe-Verlag, Berlin, 2002

[4] D.Stoffer, Variable steps for reversible integration methods, Computing 55 (1995) 1-22

[5] E.Hairer and D.Stoffer, Reversible long term integration with variable stepsizes, SIAM J. Sci. Computing, 18, (1997) 257-269

[6] M.P.Calvo and E.Hairer, Accurate long term integration of dynamical systems, Applied Numerical Mathematics, 18 (1995) 95-105

[7] D. Stoffer private communication

[8] J.M.Sanz-Serna and M.P.Calvo, Numerical Hamiltonian Problems, Applied Mathematics and Mathematical Computation, Chapman and Hall London, 1994

[9] J.M.Sanz-Serna, M.A.Lopez-Marcus and M.P.Calvo, Variable Step Implementation of geometric integrators, Applied Numerical Mathematics (1998) 1-16

[10] E.Hairer and C.Lubich, Symmetric Multistep Methods over long times, Numerical Mathematics 97 (2004), 699-723

[11] M.P.Calvo and J.M.Sanz-Serna, Variable steps for symplectic integrators in Numerical Analysis, 1991, Pitman Research Notes Math. Series 260, Longman Harlow VIC 1992 pp34-48

[12] R.I.McLachlan, G.R.W.Quispel and G.S.Turner, Numerical integrators that preserve symmetries and reversing symmetries, SIAM J. Numer. Anal., 35 (1998), pp. 586-599