Generic Euler extrapolations. Accumulation errors apparently independent of precision of floating point arithmetic

C.C. Lalescu[a,b1]

[a] *Statistical and Plasma Physics, Université Libre de Bruxelles, Campus Plaine, CP 231, B-1050 Brussels, Belgium*

[b] *Faculty of Physics, University of Craiova, A.I. Cuza 13, 200585 Craiova, Romania*

Abstract

A particular class of numerical solvers for generic first oder ODEs is constructed. Generally integration errors for integration schemes show a specific behaviour (due to the use of finite precision arithmetic) for small enough time steps. It is shown with numerical experiments how this class of numerical solvers behaves in an unexpected way.

1 Introduction

A general first order differential equation in an n-dimensional euclidian space is considered:

$$\frac{d}{dt}x = v(x). \tag{1}$$

If it exists, let the solution to this differential equation be given formally by ξ:

$$x(t+h) = \xi(x(t), h) \tag{2}$$

To find a numerical approximation of $\xi(\cdot, h)$, there are several possible approaches; some examples are the well known Runge-Kutta schemes, or, if Eq (1) has certain symmetries, specialised solvers like composition schemes [1, 2, 3, 4, 5]. The simplest numerical integration scheme is the Euler method:

$$y^{(1)}(x, h) = x + hv(x) \tag{3}$$

This method is of the global order 1, meaning that the integration error for a finite time T is proportional to the timestep h used. Note that in the following only the global order of an integration scheme is refered to. In [6] it is explained that the Euler scheme can be used to produce a second order approximation through extrapolation. Here that idea is generalized into a class of Euler extrapolation (EE) schemes; previous such classes of Euler extrapolations are discussed in [7].

This paper is structured in two main parts: first a presentation of the algorithm is given in the next section, with the overview of the rigurous construction following. Numerical experiments are then discussed: first it is shown that these extrapolations are generally problematic in the context of floating point arithmetic. Specific "good" extrapolations are then presented, and the method of finding them is discussed. In the end, a brief comparison with symplectic composition methods is performed.

[1] *e-mail:* clalescu@ulb.ac.be

2 Algorithm

The method is first presented here fully for clarity without proof. Let us note the "k applications of the Euler scheme" with

$$e(x, v, h, 1) \equiv x + hv(x) \tag{4}$$
$$e(x, v, h, k) \equiv e(e(x, v, h, k-1), v, h, k) \tag{5}$$

A sequence of n positive integers is chosen:

$$K = \{k_i\}_{1 \leq i \leq n} \text{ with } k_i < k_j \ \forall i < j \tag{6}$$

The "coefficient function" is computed as:

$$c(K, i) = k_i^{n-1} \prod_{\substack{j \neq i \\ 1 \leq j \leq n}} \frac{1}{k_i - k_j} \tag{7}$$

Then, it will be shown in section 3 that the following formula gives an nth order integration scheme:

$$\xi^{(K)}(x, v, h) = \sum_{i=1}^{n} c(K, i) e(x, v, h/k_i, k_i) \tag{8}$$

3 Order proof

In a more general setting, consider an increasing sequence of natural numbers

$$f : \mathbb{N}^* \to \mathbb{N}^*, \ f(n) < f(m) \ \forall n < m \tag{9}$$

With the above notation of recursive applications of the Euler scheme, let

$$z^{(f,n)}(h) = e(x, v, h/f(n), f(n)) \tag{10}$$

Because the Euler scheme is a first order scheme, the following hold for the errors of the z values:

$$z^{(f,1)} = \xi + \epsilon_1 \frac{h}{f(1)} + \epsilon_2 \left(\frac{h}{f(1)}\right)^2 + \cdots + \epsilon_k \left(\frac{h}{f(1)}\right)^k + \ldots \tag{11}$$
$$z^{(f,2)} = \xi + \epsilon_1 \frac{h}{f(2)} + \epsilon_2 \left(\frac{h}{f(2)}\right)^2 + \cdots + \epsilon_k \left(\frac{h}{f(2)}\right)^k + \ldots \tag{12}$$
$$\ldots \tag{13}$$
$$z^{(f,n)} = \xi + \epsilon_1 \frac{h}{f(n)} + \epsilon_2 \left(\frac{h}{f(n)}\right)^2 + \cdots + \epsilon_k \left(\frac{h}{f(n)}\right)^k + \ldots \tag{14}$$
$$\ldots \tag{15}$$

From this set of relations, extract the first n, and just keep the first n terms on the right hand side. This results in a linear system of equations, with the unknowns ξ and ϵ_k. It will be shown that the resulting "solution" for ξ is the $\xi^{(F)}$ formula, where $F = \{f(k)\}_{1 \leq k \leq n}$.

Because this is a linear system of equations, the solution is a ratio of determinants. With a slight abuse of notations, let:

$$D(f,n,z) = \begin{vmatrix} z^{(f,1)} & \frac{1}{f(1)} & \frac{1}{f(1)^2} & \cdots & \frac{1}{f(1)^{n-1}} \\ z^{(f,2)} & \frac{1}{f(2)} & \frac{1}{f(2)^2} & \cdots & \frac{1}{f(2)^{n-1}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ z^{(f,n)} & \frac{1}{f(n)} & \frac{1}{f(n)^2} & \cdots & \frac{1}{f(n)^{n-1}} \end{vmatrix} \tag{16}$$

and then the solution is

$$\xi^{(F)} = \frac{D(f,n,z)}{D(f,n,1)} \tag{17}$$

The h was omitted from the D notation because it would simplify anyway in the formula for ξ. Note, however, that $D(f,n,1)$ is a Vandermonde determinant:

$$D(f,n,1) = \prod_{1 \le i < j \le n} \left(\frac{1}{f(j)} - \frac{1}{f(i)} \right) \tag{18}$$

Similarly, if $D(f,n,z)$ is expanded after the first column, smaller Vandermonde determinants are found:

$$d(f,n,i) = \left(\prod_{\substack{j \neq i \\ 1 \le j \le n}} \frac{1}{f(j)} \right) \prod_{\substack{k_1,k_2 \neq i \\ 1 \le k_1 < k_2 \le n}} \left(\frac{1}{f(k_2)} - \frac{1}{f(k_1)} \right) \tag{19}$$

Then $\xi^{(F)}$ can be rewritten

$$\xi^{(F)} = \frac{1}{D(f,n,1)} \sum_{1 \le i \le n} (-1)^{i+1} z^{(f,i)} d(f,n,i) \tag{20}$$

The coefficient for each $z^{(f,i)}$ is:

$$c(f,n,i) = \frac{(-1)^{i+1}}{D(f,n,1)} d(f,n,i) \tag{21}$$

which simplifies to the value given in the algorithm:

$$c(f,n,i) = \prod_{\substack{k \neq i \\ 1 \le k \le n}} \frac{f(i)}{f(i) - f(k)} \tag{22}$$

4 Numerical experiments

The algorithm presented was constructed out of the need of an explicit high order solver for a generic ordinary differential equation. The interest was to have several approaches to the same problem, mainly for testing purposes. Outside the scope of that initial problem, it appears that this class of schemes has an interesting property in a pure numerical analysis context.

4.1 Setup

Consider an initial condition x_0 for equation (1), and some generic integration scheme $\eta(x, v, h, k)$ (same recursive definition as for e). There are at least two types of errors that can be constructed right away for the trajectory leaving from x_0:

$$\varepsilon^{(1)}(x_0, v, h, n) = \left\| \eta\left(\eta\left(x_0, v, \tfrac{h}{n}, n\right), v, -\tfrac{h}{n}, n\right)\right\| \tag{23}$$

$$\varepsilon^{(2)}(x_0, v, h, n, m) = \left\| \eta\left(x_0, v, \tfrac{h}{n}, n\right) - \eta\left(x_0, v, \tfrac{h}{m}, m\right)\right\| \tag{24}$$

$\varepsilon^{(1)}$ seems the most appropriate error computation; however, it fails to properly estimate the integration error when the system (i.e. v) has some particular symmetry, or if the map η is exactly time reversible (as is the case for widely used composition methods).

$\varepsilon^{(2)}$ should generally be used with geometric progressions (e.g. $n = 2^p$ and $m = 2^{p+1}$). It usually works just as well with more general series of steps.

As a sidenote, both ε are measurements of local integration errors — they only diagnose the accuracy of $x(t+h)$ estimated from $x(t)$ for relatively small h. In the context of particle transport, different errors can be much more important [5].

4.2 Error behaviour for EE

For the sake of simplicity, a generic notation for an average error is introduced:

$$\varepsilon(p) = \langle \varepsilon^{(1)}(x_i, v, 1, 2^p)\rangle_i \tag{25}$$

x_i is assumed to be a representative set of initial conditions for the system considered, and $\varepsilon(p)$ is the average error over the x_i.

For any properly implemented η of global order n, when p increases enough, the error is proportional to the nth power of the timestep $\varepsilon(p) \sim 2^{-np}$. However, finite precision arithmetic leads to the accumulation of round off errors in the result. So once 2^p becomes large enough, $\varepsilon(p)$ no longer decreases (it usually starts to increase proportionally to 2^p). This behaviour is well known [8], and quite natural. If more accurate results are required, correspondingly higher precision arithmetic is needed in the intermediary computations.

It appears that this error saturation mechanism is different for the $\xi^{(K)}$ schemes. Figures 1, 2 and 3 show integration errors for a 2 dimensional chaotic system defined by

$$\begin{aligned} H(x, y, p_x, p_y) &= \tfrac{1}{2}(p_x^2 + p_y^2) \\ &+\tfrac{1}{2}\cos(x) + \tfrac{1}{2}\cos(y) + \tfrac{1}{4}\cos(x)\cos(y) \end{aligned} \tag{26}$$

(the initial conditions were $x = y = 0$, and a randomly oriented momentum of 1)[2]. Note that the alternative function $\varepsilon(\Delta t)$ is used for the graphical representations.

If the floating point precision is changed from double to quadruple, errors obtained with solvers constructed with some specific K do not change. Extensive tests show that this behaviour is quite generic, and the relationship between the error curves obtained with different sets of integers stays the same if the system considered is changed (1D physical pendulum, 1D harmonic oscillator, etc).

[2]The font used for the plots is tiny so that the titles do not fill out the entire plot. The important message is that there are curves that look identical in the center and right plots.

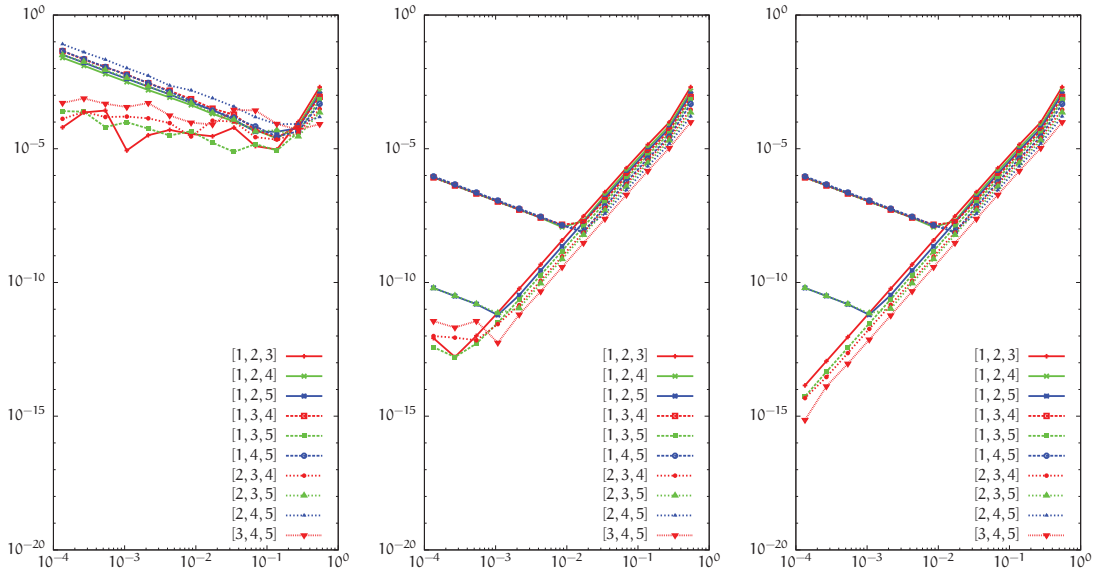

Figure 1: Integration error for a few 3rd order ξ (single precision left, double center and quadruple on the right).

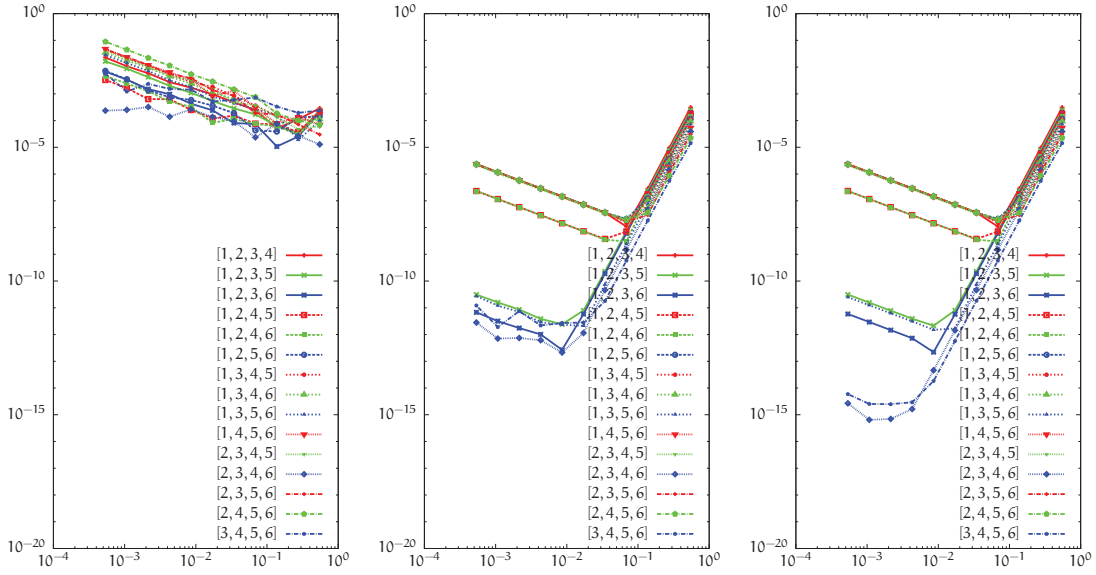

Figure 2: Integration error for a few 4th order ξ (single precision left, double center and quadruple on the right).

4.3 Finding the "good" EE

Naturally, the results in the previous section demand an explanation.

The coefficients $c(K, i)$ have alternating signs, and they generally have a wide range in absolute value for any given K. Runge Kutta schemes, for example, also contain weighted averages, but all the weights are positive (and thus they are all subunitary); also, the weights are usually on the same scale. It is therefore not surprising that the EE

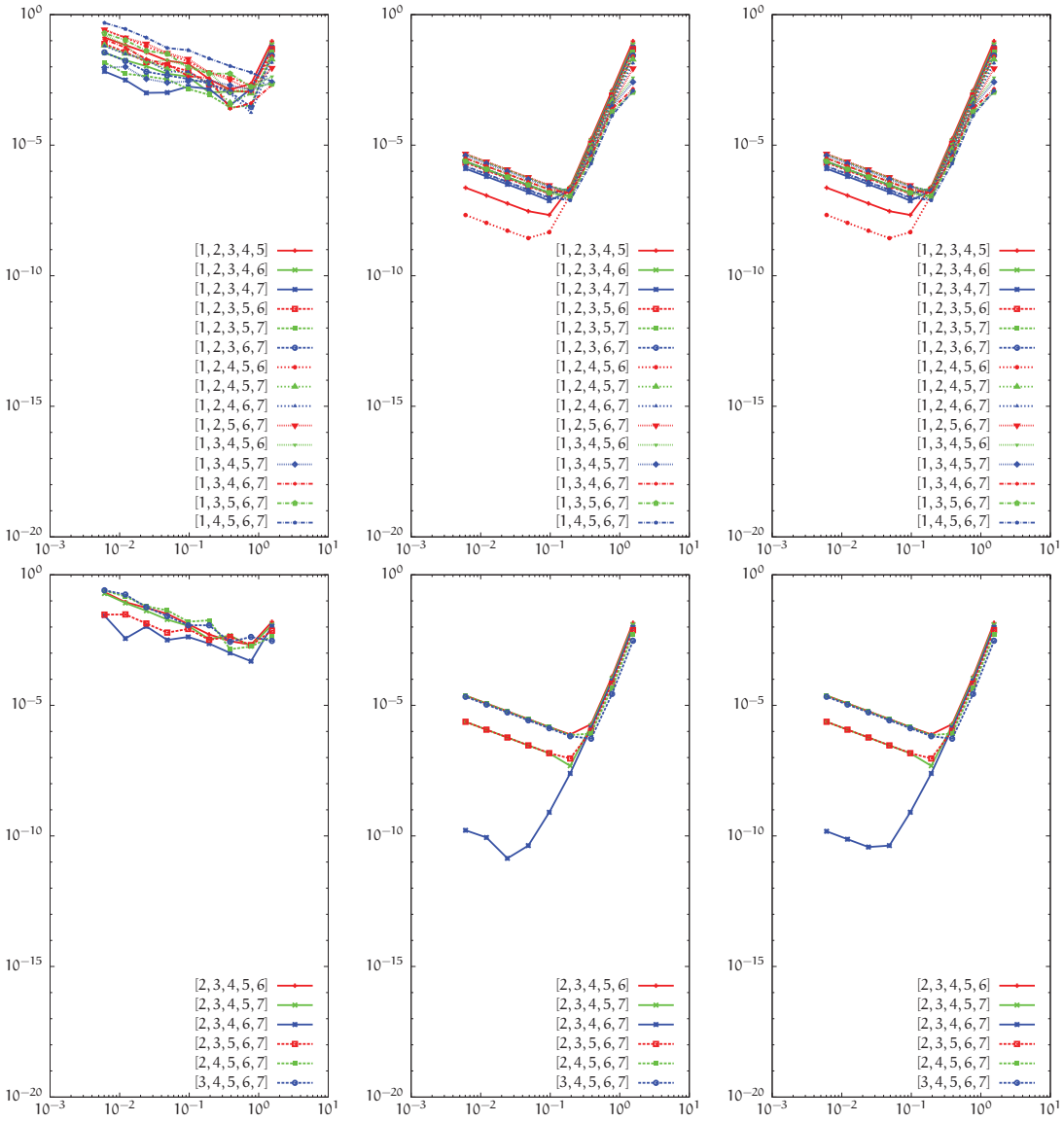

Figure 3: Integration error for a few 5th order ξ (single precision left, double center and quadruple on the right).

schemes are "bad" in a certain sense (the minimum error is not very small). However, it is surprising that the minimum error does not decrease with the improvement of floating point precision. Also, there is no obvious reason for the presence of "good" EE schemes (i.e. those for which the minimum error does change). Note that this work is limited to the comparison of single, double and quadruple precision computations; systematic investigations of higher precision results are left for the future.

One interesting question that arises is how to construct the sets K for which $\xi^{(K)}$ is a good scheme. The best method so far has been to simply compute the errors, and choose the good K sets after the fact. This approach is reasonable for low order schemes, but is very costly for card$K \geq 8$.

In order to find good sets K efficiently, a physical pendulum

$$H(q,p) = \frac{1}{2}p^2 - \cos(q) \tag{27}$$

is considered, with one fixed initial condition ($q = 1$, $p = 1$ seemed as appropriate as any other).

Arrays of K sets are then constructed. Several methods were employed:

1. a cardinal o is considered, and an offset o_e (the o stands for order of the solver). Combinations of o elements from $\{1, 2, \ldots, o + o_e\}$ are constructed systematically, and error plots such as those in figure 1 are generated. By combing through the resulting plots, good sets K are chosen. This is reasonable in terms of CPU time for $o \leq 6$. For larger orders, the offset o_e required to reach good schemes is very large, and the number of K sets becomes too large. Figures 1, 2 and 3 are a small example of this approach ($o_e = 2$).

2. Combinations are constructed as before, but only the error for a fixed $p \sim 10$ is computed. An array containing triplets (K counter from the list of combinations, error and K itself) is output into two text files, sorted by id (counter), and another array is sorted by error. The first few results from the list sorted by error can be tested. The results of the id sorted list are plotted to look for some empiric first filter for the K sets — see figure 4.

3. After using the second method, it was noticed that most good K begin with a series of consecutive numbers. This results in the first rough filter: choose o and o_e; choose i; for $j < o_e - o$, add $\{j, j+1, \ldots, j+i\}$ to combinations of $\{j+i+1, \ldots, o+o_e\}$. With this filter, it was possible to find good 8th order EE schemes.

It is likely that some good schemes are missed by using the empiric filter, but the total number of combinations tested must remain manageable. Note that the filter was inspired by a plot such as figure 4, but for $(o, o_e) = (8, 17)$; the correspondingly larger datasets can be provided on request.

4.4 Calibration with CM

The first hint that the EE schemes are problematic for small time steps was given when they were compared with symplectic Composition Methods. In this subsection, the generic error considered is based on ε^2:

$$\varepsilon(p) = \langle \varepsilon^{(2)}(x_i, v, 1, 2^p, 2^{p+1}) \rangle_i \tag{28}$$

A set K is "good" if the corresponding $\xi^{(K)}$ shows similar behaviours of $\varepsilon(p)$ for large p. The same 2D system as before is used to generate figure 5. Only even orders for the CM are considered (with CM coefficients recommended in [1]).

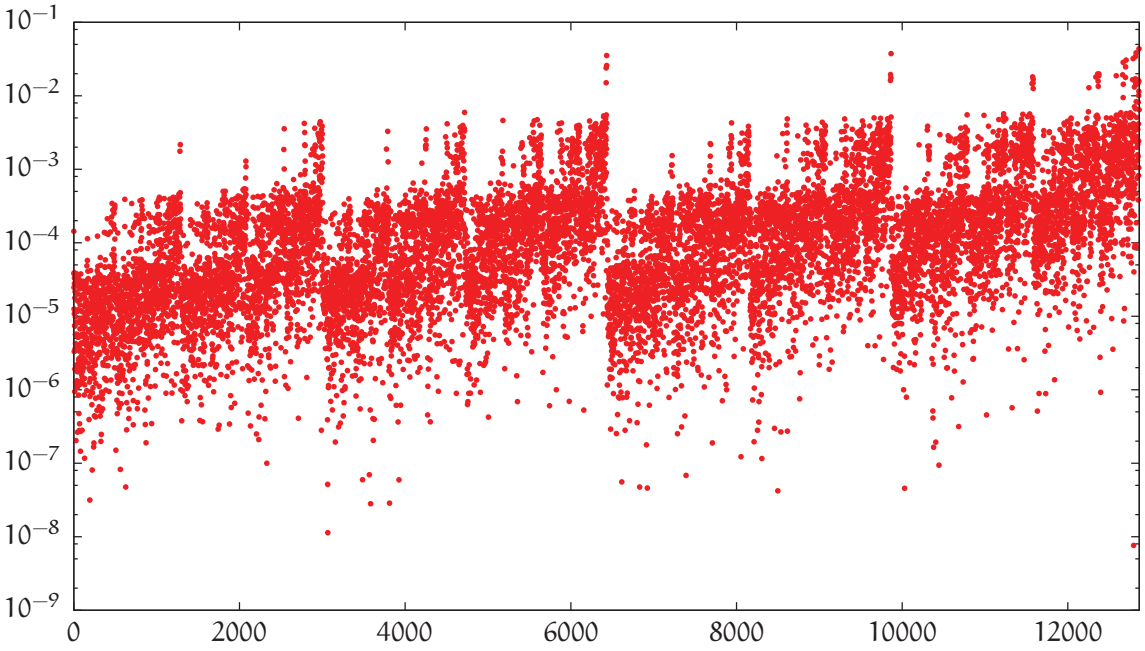

Figure 4: Integration errors $\varepsilon(11)$ for $(o, o_e) = (8, 8)$ versus the K id. Statistically, "discontinuities" are observed where k_1 is changed, and further, smaller discontinuities, where k_2, ... are changed.

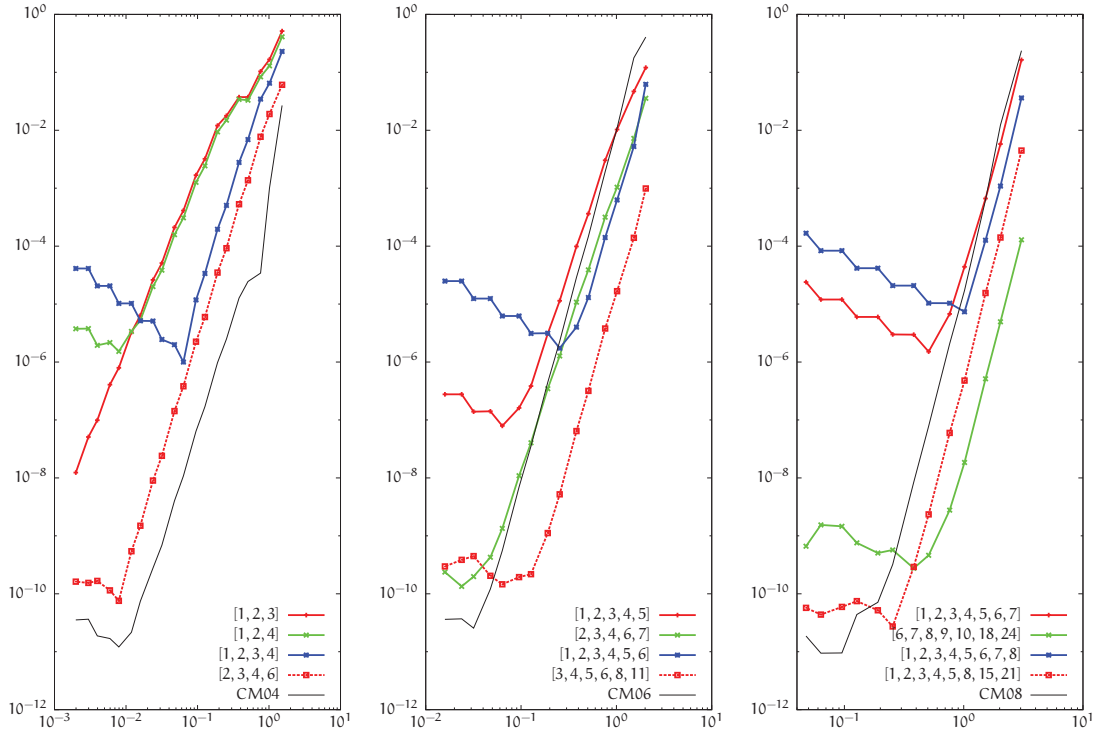

Figure 5: Comparison of EE and CM. Computations were performed in double precision.

5 Predicting error behaviour

Once the error behavior became apparent, the generating mechanisms were probed. The most direct approach is to simply test if there are any systematic errors. Consider the system of equations used to find the extrapolation formula. It is possible to compute the coefficient of the term (from $\xi^{(K)}$) proportional to h^j with a simple sum:

$$s(K, j) = \sum_{1 \leq i \leq n} c(K, i) k_i^{-j} \tag{29}$$

This results in an array of coefficients for h^j error terms, and the first assumption was that these errors are doing something special for good K sets. It was assumed that $s(K, j)$ decreases very rapidly with j for good K, but it seems this is not the good diagnostic. Computations show that for all K, $s(K, j)$ either increases exponentially, reaches a stable limit or decreases exponentially. Furthermore, plots of the limit $s(K, j)/s(K, j + 1)$ are much more regular than error plots such as figure 4.

6 Discussion

A class of integration schemes, Euler extrapolations, has been presented. EE solvers can be used for generic systems, however they are extremely inefficient when compared to Runge Kutta schemes, so they are mostly of academic interest.

An interesting aspect for these schemes is their behaviour for small timesteps. This behaviour is reminiscent of the Gibbs phenomenon [9], where considering more terms of a Fourier series does not improve the behaviour around discontinuities. Only in this case increasing numerical precision does not improve errors for small time steps. However, further studies with higher precision should be performed before drawing a conclusion.

Because these schemes are solutions of a linear system of equations, it is expected that they have very good properties with respect to the conservation of invariants. In practice, the Jacobian of some specific $\xi^{(K)}$ is also the solution of a linear system of equations, and it should also differ from the Jacobian of ξ by an error of order card$K$. This would mean for instance that energy would be preserved for a conservative Hamiltonian system with a high order, unlike the classic Runge Kutta scheme where there is a first order error in the energy. This property isn't directly relevant for Hamiltonian systems, where implicit Gauss Runge Kutta schemes work perfectly even for unseparable Hamiltonians (symplectic CM are recommended for the separable case). However it becomes interesting in the case of more general volume preserving flows.

As a final note, the code used for this work is available under the GPL, and will be provided on request. It was written in Python, using C for the intensive tasks of actually integrating the equations.

Acknowledgements

Helpful discussions with Gy. Steinbrecher are greatfully acknowledged. The author would also like to thank E. Hairer for valuable comments.

References

[1] Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric Numerical Integration — Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer, 2nd edition, 2006.

[2] Haruo Yoshida. Construction of higher order symplectic integrators. *Physics Letters A*, 150:262, 1990.

[3] S. Blanes and P.C. Moan. Practical symplectic partitioned Runge-Kutta and Runge-Kutta-Nyström methods. *Journal of Computational and Applied Mathematics*, 142:313–330, 2002.

[4] Siu A. Chin. Symplectic and energy-conserving algorithms for solving magnetic field trajectories. *Physical Review E*, 77:066401, 2008.

[5] Robert I McLachlan and G Reinout W Quispel. Geometric integrators for ODEs. *Journal of Physics A: Mathematical and General*, 39(19):5251–5285, 2006.

[6] Peter E. Kloeden and Eckhard Platen. *Numerical Solution of Stochastic Differential Equations*. Springer, second corrected printing edition, 1995.

[7] E. Hairer, S. P. Norsett, and G. Wanner. *Solving Ordinary Differential Equations I. Nonstiff Problems*. Springer, second revised edition edition, 1993.

[8] Richard Fitzpatrick. *Computational Physics — lecture notes*. The University of Texas at Austin, 2006.
URL: http://farside.ph.utexas.edu/teaching/329/329.html,

[9] E. W. Weinstein. Gibbs phenomenon.
URL: http://mathworld.wolfram.com/GibbsPhenomenon.html,