

A new implementation of symplectic Runge–Kutta methods

Robert I. McLachlan*

March 2, 2006

Abstract

We propose a new iteration for solving implicit Runge–Kutta equations, using the Jacobian of the vector field and matrix–vector multiplies whose extra cost for certain structured problems is negligible. The iteration reduces the number of vector field evaluations almost to that of Newton’s method, often only 1 or 2 per time step, making symplectic Runge–Kutta methods efficient even at relatively large time steps.

1 Introduction

Symplectic integrators based on splitting, such as the leapfrog method for separable Hamiltonian systems, are fast and simple and give very good results for long simulations [2]. They are widely, almost universally used in applications from accelerator physics to molecular dynamics to celestial mechanics [5]. However, the only symplectic integrators for *general* Hamiltonian systems are implicit, such as the Gaussian Runge–Kutta methods [3], which has tended to limit their popularity. (The implicit equations must be solved to within round-off error, to preserve symplecticity.) Hairer, Lubich, and Wanner [2] have studied the implementation issues and found the simple (‘standard’) iteration, Eq. (4) below, superior to Newton’s method; for very small error tolerances it can even compete with high-order methods based on splitting and composition. But for the relatively low orders and large time steps often used in long symplectic integrations, the convergence of the standard iteration deteriorates, making the method more expensive, which defeats the purpose of using a large time step.

In this paper we propose a very simple Newton-chord iteration [7] which, for many structured problems, costs only a constant factor close to 1 times the cost of the standard iteration, yet has a convergence rate very close to Newton’s method. For the midpoint rule, for example, the method requires only 1 or 2 evaluations of the vector field per time step, making it essentially as fast as an explicit method like leapfrog.

The method uses the Jacobian of the vector field and performs matrix–vector multiplies. It is efficient when those two operations are cheap compared to the cost of evaluating the

*IFS, Massey University, Palmerston North, New Zealand. Email: r.mclachlan@massey.ac.nz

vector field itself. Standard implementations of implicit methods try to balance the number of Jacobian factorizations against the number of vector field evaluations [8]. In contrast, our goal is minimize the number of vector field evaluations without performing any operations that cost substantially (e.g., asymptotically, for large systems) more than a vector field evaluation.

First, consider N -body problems with Hamiltonians of the form

$$H = \sum_{i,j=1}^N p_i G(\|q_i - q_j\|) p_j + V(\|q_i - q_j\|). \quad (1)$$

The cost of evaluating the vector field directly is $\mathcal{O}(N^2)$. If $G^{(k)}$ and $V^{(k)}$ are cheap then the cost is dominated by the cost of computing $\|q_i - q_j\|$ and the Jacobian is available essentially for free. This has been exploited for separable systems ($G = 1$) by designing efficient high-order symplectic integrators using the Jacobian and matrix–vector multiplies [1, 5]. On the other hand, if G and V are special functions then it often happens that it is relatively cheap to compute their derivatives at the same time. (This happens in the example below, where G is an exponential or a Bessel function.) The matrix–vector multiplies, on the other hand, are also $\mathcal{O}(N^2)$ and hence carry a constant work overhead compared to the cost of evaluating the vector field.

Second, consider lattice systems with independent variables indexed by a lattice L and Hamiltonian of the form

$$H = \sum_{i \in L} F(x_{i+\Delta}) \quad (2)$$

where $\Delta \subset L$ is a fixed template specifying which sites are coupled. The cost of computing the vector field is $\mathcal{O}(N)$. The Jacobian matrix has only $\mathcal{O}(N)$ nonzero entries and hence matrix–vector multiplies cost $\mathcal{O}(N)$. Again, the cost of forming and multiplying by the Jacobian is negligible for certain functions F .

We describe the algorithm for the implicit midpoint rule. (Its extension to any implicit Runge–Kutta method is straightforward.) For the ODE $\dot{x} = f(x)$ we are required to solve

$$g(z) := z - f(x_0 + \frac{1}{2}\Delta tz) = 0 \quad (3)$$

for z , where Δt is the time step. (The final update is $x_0 \mapsto x_1 := x_0 + \Delta tz$.) The standard fixed-point iteration is

$$z^{k+1} = f(x_0 + \frac{1}{2}\Delta tz^k) = z^k - g(z^k) \quad (4)$$

and Newton’s method is

$$z^{k+1} = z^k - g'(z^k)^{-1}g(z^k) = z^k - (I - A)^{-1}g(z^k) \quad (5)$$

where $A = \frac{1}{2}\Delta t f'(z^k)$.

We replace the iteration matrix $(I - A)^{-1}$ by its Taylor series $\sum_{i=0}^M A^i$ of order M to give the

Outer iteration:

$$z^{k+1} = h(z^k) := z^k - \left(\sum_{i=0}^M A^i \right) g(z^k) \quad (6)$$

which can be evaluated using one vector field and Jacobian evaluation at z^k and M matrix-vector multiplies. It is a Newton-chord iteration using a certain approximate Jacobian.

Let z^* be the desired solution so that $g(z^*) = 0$ and let $e^k = x^k - x^*$ be the error in the current approximation. Then by Taylor's theorem we have for some $\theta \in [0, 1]$,

$$\begin{aligned} e^{k+1} &= h'(z^k)e^k + \frac{1}{2}h''(z^k + \theta(z^{k+1} - z^k))(e^k, e^k) \\ &= \left(I - \left(\sum_{i=0}^M A^i \right) (I - A) \right) e^k + \frac{1}{2}h''(z^k + \theta(z^{k+1} - z^k))(e^k, e^k) \\ &= A^{M+1}e^k + P(e^k, e^k) + \mathcal{O}(\|e^k\|^3) \end{aligned} \quad (7)$$

where $P = h''(z^*) = \mathcal{O}((\Delta t)^2)$. The convergence rate of the modified iteration (6) is therefore $\rho(A^{M+1}) = \rho(A)^{M+1} = \mathcal{O}((\Delta t)^{M+1})$ where $\rho(A)$ is the convergence rate of the standard iteration. Not only is the modified iteration higher order in the time step, but (at the linear level, i.e. for sufficiently small e^k) one step of the modified iteration is *identical* to $M + 1$ steps of the standard iteration¹. It is therefore worth increasing M by one precisely when a Jacobian-vector multiply is cheaper than evaluating the vector field.

We also conclude from (7) that the modified iteration (6) converges for sufficiently small $\|e^0\|$ if and only if the simple iteration converges. We therefore henceforth assume that $\rho(A) < 1$.

If M is taken too large, however, then e^{k+1} will be dominated by the quadratic term in (7). Note that as $M \rightarrow \infty$, $A^{M+1} \rightarrow 0$ and $P \rightarrow P^\infty$ (say), the iteration tensor for Newton's method. In fact $P = P^\infty + \mathcal{O}((\Delta t)^{M+1})$ so as $M \rightarrow \infty$ the behaviour of the modified iteration tends to that of Newton's method. One should therefore adapt M depending on the current iteration, so that $\|A^{M+1}e^k\| \approx \|P(e^k, e^k)\|$. This gives the following algorithm in which (6) is implemented by the following

Inner iteration:

1. Given z^k , set $w^0 = g(z^k)$;
2. While $\|w^{m+1} - w^m\| > \max(c\|w^0\|^2, tol)$, set $w^{m+1} = w^0 + Aw^m$;
3. Set $z^{k+1} = z^k - w^m$.

¹Often, higher-order methods have large error constants, rendering them useless unless the time step is sufficiently small. This does not happen here.

Here tol is the tolerance, usually close to machine precision, to which the Runge–Kutta equations are to be solved. The parameter c depends on the problem and is related to the norm of P , i.e. to the second derivatives of the vector field.

This algorithm will lead to the number M of inner iterations roughly doubling each iteration, just as the error decreases in Newton’s method.

It is possible, as in the automatic detection of stiffness [3] and the traditional convergence tests for implicit methods [8], to maintain running estimates of the slowly-varying parameters $\rho(A)$ and the norm of P , and hence adapt the choice of M more precisely. We have not pursued this approach because (i) our goal is a simple algorithm; (ii) in experiments, varying c by a factor of 10^4 hardly affected the resulting efficiency; and (iii) it is not that easy to estimate $\rho(A)$ when A is non-normal and the convergence of the inner iteration in the chosen norm is erratic. (In theory, one should choose $c = \mathcal{O}((\Delta t)^2)$ to reflect the faster convergence of the Newton iterates as $\Delta t \rightarrow 0$; but this also seemed to make no difference in practice.) Our experiments reported below use $c = 1$ and $tol = 10^{-15}$ and $\|\cdot\|_\infty$ error norms.

To further reduce the number of evaluations of f we terminate the outer iteration (6) at z^{k+1} when $\|g(z^k)\| < tol_2$ where $tol_2 > tol$. Under the model used for terminating the inner iterations, one reasonable choice is to take $tol_2 = \sqrt{tol}/c$, as the final iteration will then take the error down to about $2tol$. This saves one evaluation of f and is the termination criterion used in our experiments. If long-term growth of round-off error is a concern, these criteria may need to be tightened.

A good initial guess is essential for the performance of this method as one wants the underlying Newton’s method to stay close to the root where convergence is quickest. We have used polynomial extrapolation from the previous time steps [2]. The order s of extrapolation is adapted based on the decrease of the backward differences:

Initialization iteration:

1. Set $w^0 = z_n$;
2. While $\|\nabla^{s+1} z_n\| < \|\nabla^s z_n\|$, set $w^{s+1} = w^s + \nabla^{s+1} z_n$;
3. Set $z_{n+1}^0 = w^s$.

The modified iteration (6) can be further sped up by freezing the Jacobian for several iterations or for several time steps. Here we have only tested the basic method, the initialization, outer, and inner iterations, which together are about as simple as possible while still much faster than the standard iteration.

Example. The following example arose in a study of the dynamics of certain curves called momentum sheets in a generalized Camassa–Holm equation that is thought to govern certain classes of solitary waves in shallow water [4]. The curve motion is governed by a PDE which is discretized by placing particles at positions q_i on the curve. Each particle

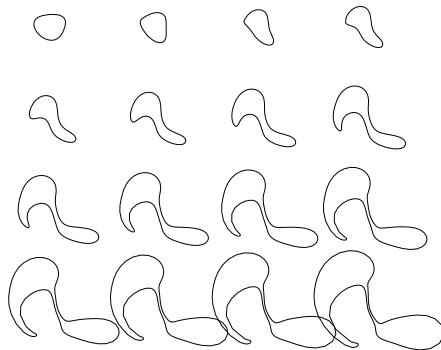


Figure 1: Snapshots of the motion of the curve in the example, for $t \in [0, 50]$.

carries momentum p_i and the Hamiltonian is given by Eq. (1) with $V(r) = 0$ —that is, the Hamiltonian describes geodesics on a Riemannian manifold with a metric the inverse of the matrix of G s. The three kernels G of most interest in the application were the Gaussian $G_1(r) = \exp(-r^2)$, the modified Bessel function $G_2(r) = K_0(r)$ (with the singularity at $r = 0$ appropriately removed), and $G_3(r) = rK_1(r)$. Note that we have, e.g., $G_2'(r) = -K_1(r)$ and $G_2''(r) = (K_0(r) + K_2(r))/2$, which can be efficiently computed with relatively little overhead compared to the cost of $K_0(r)$ itself.

We have used the simplest kernel $G_1(r)$ and computed the motion of the sheet shown in Figure 1 for time 50, during which it undergoes a large-scale evolution. There are 100 particles and the total dimension of the system is 400. To give an idea of the time scale of this problem, both the simple and modified iterations converge at $\Delta t = 6$ but not at $\Delta t = 7$. The final position error is 0.7% at $\Delta t = 1$ and 15.3% at $\Delta t = 5$. The maximum relative energy error during the simulation is 0.2% at $\Delta t = 1$.

The results are shown in Table 1 and Figure 2. The number of outer iterations (equivalently, number of vector field evaluations) per time step is very close to that used by Newton’s method. Roughly, one could use $\Delta t = 0.4$ and do about 1 evaluation of f per step (the standard iteration needs 6), achieving a final position error of 0.0011; or one could use $\Delta t = 0.8$ and do about 2 evaluations of f per step (the standard iteration needs 10), achieving a final position error of 0.0045. Clearly, in this problem there is no advantage in the larger time step. But even $\Delta t = 0.4$ is relatively large for this problem and is analogous to the large time steps often used in molecular dynamics problems.

A second observation is that even with the simplest implementation possible, namely using $c = 1$ in the termination criterion for the inner iteration, we have $N_f + N_{\text{inner}} \approx N_{f,\text{std}}$ across the whole range of time steps. The algorithm is roughly equivalent to exchanging

N_{steps}	Δt	N_f	N_{inner}	s	$N_{f,\text{std}}$
10	5	4.60	39.20	3.30	42.10
14	3.57	3.64	24.78	3.71	28.35
20	2.50	3.20	17.15	4.30	20.65
28	1.78	3.00	13.85	5.32	16.39
40	1.25	2.50	10.72	6.80	13.00
57	0.87	2.05	8.40	8.89	10.40
80	0.62	1.97	6.45	10.78	8.20
113	0.44	1.35	4.82	13.46	6.03
160	0.31	1.05	3.25	15.41	4.29
226	0.22	1.02	2.24	14.49	3.27
320	0.15	1.01	2.02	12.51	3.06

Table 1: The time step, initially 5, is successively reduced by a factor of $\sqrt{2}$ and adjusted so that $N_{\text{steps}} = 50/\Delta t$ is an integer. N_f is the mean number of vector field evaluations per time step averaged over the run used by the modified iteration Eq. (6) (including the initial steps, when the order of the initialization is necessarily lower and more iterations are needed). N_{inner} is the mean number of inner iterations (= number of matrix–vector multiplies) per time step. s is the mean order of the extrapolation used to obtain an initial guess. $N_{f,\text{std}}$ is the mean number of vector field evaluations per time step used by the standard iteration (4).

most iterates of the standard iteration (4) with a matrix–vector multiply.

The modified iteration also works successfully for higher-order Runge–Kutta methods. With an s stage method, the need for $sN \times sN$ matrix–vector multiplies can be avoided by diagonalizing the matrix (a_{ij}) of Runge–Kutta coefficients, as in standard implementations [2]. Table 2 gives the results for the same problem under the 4th order, 2 stage Gaussian Runge–Kutta. The position errors are now only 0.2% at $\Delta t = 5$ and 4×10^{-6} at $\Delta t = 1$, and the maximum relative energy errors 0.3% at $\Delta t = 5$ and 3×10^{-5} at $\Delta t = 1$. The improvements to the 4th order method are very similar to those enjoyed by the 2nd order method. The number of outer iterations is reduced slightly, and the number of inner iterations is significantly reduced at large time steps, because the spectral radius of (a_{ij}) is reduced by a factor 0.58. For the same reason, the 4th order method converges for larger time steps, failing to converge only at $\Delta t = 9$. For this problem, the 4th order method always gives a smaller error for a given work (see Figure 3). Both methods can be used satisfactorily at large time steps.

Conclusions. Even if the vector field has no particular structure, the modified iteration (6) is likely to be an improvement over the standard iteration (4), especially if one takes the trouble to update the Jacobian only when required. The only unavoidable requirement

N_{steps}	Δt	N_f	N_{inner}	s
6	8.33	4.67	36.16	1.83
10	5	4	21.90	2.50
14	3.57	3.50	16.71	2.71
20	2.50	3	13.05	3.65
28	1.78	2.96	10.85	4.46
40	1.25	2.45	8.90	6.20
57	0.87	2.05	7.40	7.75
80	0.62	2.02	5.85	10.20
113	0.44	1.46	4.69	13.11
160	0.31	1.05	3.33	15.76
226	0.22	1.02	2.48	14.92
320	0.15	1.01	1.90	12.76

Table 2: As for Table 1, but using 4th order Gaussian Runge–Kutta.

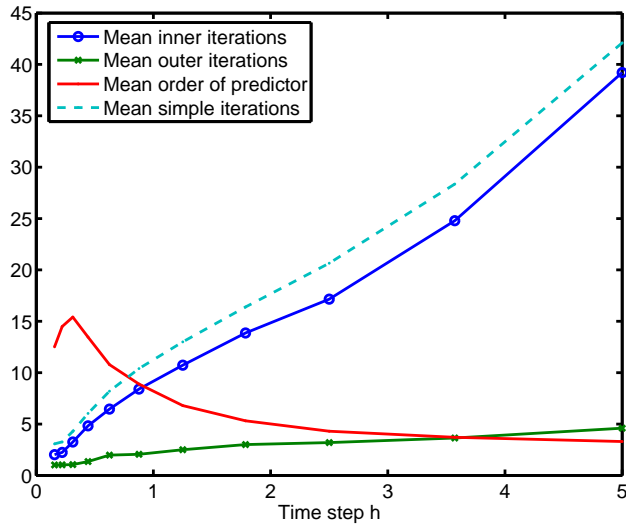


Figure 2: Results of the modified iteration applied to the midpoint rule in the Example.

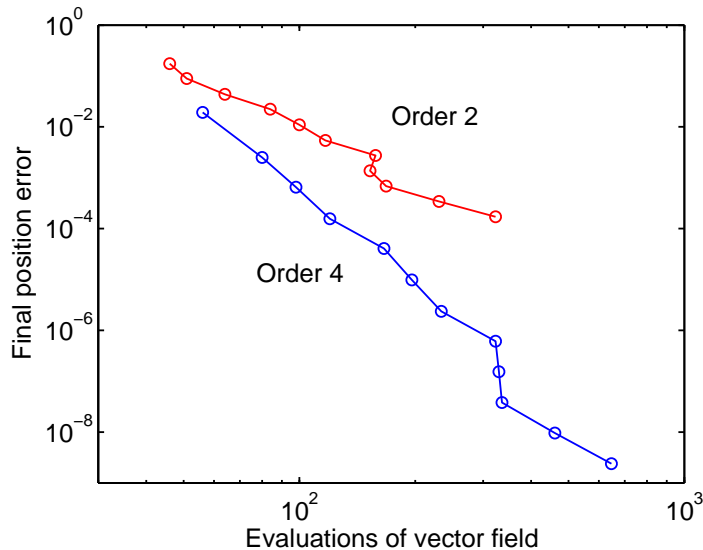


Figure 3: 2nd and 4th order Gaussian Runge–Kuttas compared for the Example.

is that we need $\rho(A) < 1$, i.e., the problem must not be stiff. But the case $\rho(A) > 1$ takes us into the entirely different realm of highly oscillatory systems [2]. Therefore the present algorithm is most suited to nonstiff problems in which implicit methods are needed for their geometric properties such as symplecticity.

Many large physical systems have an appropriate structure which reduces the cost of the integration considerably, down to one or two vector field evaluations per time step. Of course, some systems have so much structure that it is possible to exploit it directly in solving the implicit Runge–Kutta equations. For example, for N -body systems one could use a fast multipole method and for smooth lattice systems such as those arising as semidiscretizations of PDEs one could use multigrid. However, these approaches are not only vastly more complicated than the present proposal, they are also much more problem specific.

It is tempting to ask if the information contained in the Jacobians can be used for other purposes than just speeding up the convergence. Unfortunately, there are no symplectic multiderivative Runge–Kutta methods [2], nor are there any “elementary differential” Runge–Kutta methods using just the Jacobian—the simplest is the midpoint rule applied to the modified Hamiltonian $H + \frac{1}{24}(\Delta t)^2 f^T H'' f$, which requires the second derivative of the vector field f . There are Runge–Kutta-like methods using f' [6], but these are not very developed.

Hairer, Lubich, and Wanner [2] have already reported experiments in which the implementation of Gaussian Runge–Kutta methods by a Gauss–Seidel iteration (with conver-

gence rate $\mathcal{O}((\Delta t)^2)$) is already superior to an 8th order composition method for separable problems when the error tolerance is small (around 10^{-8}). It is tempting to ask if the iteration (6) can improve the performance of implicit methods still further and push their applicability to larger time steps, even for separable problems.

Acknowledgements. I am grateful to the Marsden Fund and the New Zealand Institute of Mathematics and its Applications for financial support, and to Stephen Marsland for stimulating discussions on the momentum sheet example.

References

- [1] S Blanes and P C Moan, Practical symplectic partitioned Runge–Kutta and Runge–Kutta–Nyström methods, *J. Comput. Appl. Math.* **142** (2002), 313–330.
- [2] E Hairer, C Lubich, and G Wanner, *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, Springer, Berlin, 2002.
- [3] E Hairer and G Wanner, *Solving ordinary differential equations. II. Stiff and Differential-Algebraic Problems*, 2nd ed., Springer, Berlin, 1996.
- [4] R I McLachlan and S R Marsland, The Kelvin–Helmholtz instability of momentum sheets in the Euler equations for planar diffeomorphisms, preprint.
- [5] R I McLachlan and G R W Quispel, Geometric integrators for ODEs, *J Phys A*, to appear.
- [6] S Miesbach and H J Pesch, Symplectic phase flow approximation for the numerical integration of canonical systems, *Numer. Math.* **61** (1992), 501–521.
- [7] R Schreiber and H B Keller, Driven cavity flows by efficient numerical techniques, *J. Comput. Phys.* **49** (1983), 310–333.
- [8] L F Shampine, Implementation of implicit formulas for the solution of ODEs, *SIAM J. Sci. Statist. Comput.* **1** (1980), 103–118.