

Bootstrapping Discrete–Gradient Integral–Preserving Integrators to Fourth Order

D.I. McLaren[†] and G.R.W. Quispel[‡]

[†] Department of Mathematics, La Trobe University, Victoria
3086, Australia. email: D.McLaren@latrobe.edu.au

[‡] Department of Mathematics and Centre of Excellence for Mathematics and Statistics
of Complex Systems, La Trobe University, Victoria
3086, Australia. email: R.Quispel@latrobe.edu.au

Abstract: Ordinary differential equations having a first integral may be solved numerically using one of several methods, with the integral preserved to machine accuracy. One such method is the discrete gradient method. It is shown here that the order of the discrete gradient method can be bootstrapped repeatedly to higher orders of accuracy. The potential for improved efficiency offered by the bootstrapped method is illustrated using three 6-dimensional systems.

1 Introduction

Many systems of differential equations possess so-called geometric properties, for example one or more first integrals, symplectic structure, volume preservation, and others. Numerical solution of such systems is usually best effected using a geometric integrator, i.e. one that can exactly preserve this property (or properties). There are many excellent expositions of geometric integration in the literature [2, 7, 13, 17–19, 27]. For more specialised papers, see [10, 14, 20, 23, 29].

In this paper we are particularly interested in integral-preserving integrators (IPI's) which are designed to provide exact preservation of any first integral possessed by a system of ordinary differential equations. We will have in mind in particular the preservation of integrals other than energy, as very recent methods exist that are only applicable to preserving energy in (canonical) Hamiltonian systems [25]. In the next section we give a very brief survey of available IPI's before introducing the new work that is the principal topic of this paper, namely a potentially more efficient way of implementing Discrete Gradient IPI's. This work will consider the preservation of a single first integral, leaving the preservation of more than one integral to be studied in a later publication.

The preservation of first integral(s) is important for the stability of an integrator, particularly in the case of low-dimensional and compact level surfaces. Even in the case of a symplectic system better stability may result from the use of an integral-preserving integrator, rather than one that preserves symplectic structure [5]. Furthermore, integral preservation in some cases seems to offer the possibility of linear error growth¹, better than the quadratic growth resulting from the application of other integrators to a non-symplectic system.

2 Background

An autonomous ordinary differential equation with a first integral $I(x)$ has the form

$$\frac{dx}{dt} = f(x), \text{ with } f(x) \cdot \nabla I(x) = 0, \quad x \in \mathbb{R}^n. \quad (1)$$

Theorem 1 [16, 26] *Let $I \in C^{r+1}(\mathbb{R}^n, \mathbb{R})$ be an integral of the vector field $f \in C^r(\mathbb{R}^n, \mathbb{R}^n)$, $r \geq 1$, $n > 1$, i.e. $f \cdot \nabla I = 0$. Then there exists a skew matrix function S , C^r on the domain $\{x : \nabla I \neq 0\}$, such that $f = S\nabla I$.*

Hence (1) can be written² in the form

$$\frac{dx}{dt} = S(x) \cdot \nabla I(x), \quad x \in \mathbb{R}^n \quad (2)$$

where S is a skew-symmetric $n \times n$ matrix.

A brief review of IPI's

(a) Standard projection methods [3, 7, 8, 28]: The numerical solution is constrained to remain on the invariant manifold, e.g. using Lagrange multipliers. These methods are not symmetric - i.e. not “self-adjoint”, so do not generally possess as good long-time behaviour as self-adjoint methods.

(b) Splitting methods: Quispel and Capel [24] showed that the vector field f can be split into 2-dimensional fields, each having the original integral. These are integrated exactly and finally a composition gives the solution of the original system with the integral preserved exactly.

(c) Standard discrete gradient (DG) methods and symmetric DG methods [4, 11, 16, 26] will be described later. There, higher orders are gained with a composition method [7, 15, 22, 30, 32], whereas the main point of this paper is to reach higher orders without it.

(d) Symmetric projection methods [1, 6, 7]: In standard projection methods the projection step is taken at the end of the integration, hence the lack of symmetry. The use of an “inverse projection” [7] at the start of each integration step allows the possibility of overall symmetry and self-adjointness, advantageous for long-term integration.

¹In separate computations we have observed linear error growth for the systems studied here. In these computations we destroyed the symmetry of the SA4 integrators by composition, thus isolating the integral-preservation as the cause of the linear error growth.

²under some technical conditions that are generically satisfied (see [16])

(e) Bootstrapped DG method [21] : Higher order IPI's without using a composition method were presented in that paper for the case of constant skew-symmetric matrix S .

The work presented in this paper represents a significant extension of that given in [21], principally because it caters for systems having non-constant skew-matrix S and thus looks beyond systems that are necessarily either Hamiltonian or Poisson³. Furthermore, in contrast to [21], the correction terms detailed here are themselves explicitly skew-symmetric and thus integral-preserving.

Below, we will give numerical applications of our method both to systems that are Hamiltonian/Poisson and to systems that are not.

In this paper, superscript indices will take their usual meaning as labels of vector, matrix or tensor components, and subscript indices⁴ will indicate differentiation with respect to one of the variables x_i . For example $I_i := \frac{\partial I}{\partial x_i}$, $I_{ij} := \frac{\partial^2 I}{\partial x_i \partial x_j}$, $S_k^{ij} = \frac{\partial}{\partial x_k} S^{ij}$, etc. Further, a repeated index in an expression will imply summation over that index.

3 Discrete gradients

Definition 1 For a differentiable function $I(x)$, $\bar{\nabla}I(x, x')$ is a discrete gradient (DG) of I if it is continuous and

$$\begin{cases} \bar{\nabla}I(x, x') \cdot (x' - x) = I(x') - I(x) \\ \bar{\nabla}I(x, x) = \nabla I(x). \end{cases} \quad (3)$$

Examples:

(i) Mean Value DG ([9])

$$\bar{\nabla}_{MV}I(x, x') := \int_0^1 \nabla I((1 - \xi)x + \xi x') d\xi, \quad x \neq x'$$

(ii) Midpoint DG ([4])

$$\begin{aligned} \bar{\nabla}_{MP}I(x, x') &:= \nabla I\left(\frac{1}{2}(x + x')\right) \\ &+ \frac{I(x') - I(x) - \nabla I\left(\frac{1}{2}(x + x')\right) \cdot (x' - x)}{\|x' - x\|^2} (x' - x) \end{aligned}$$

(iii) Coordinate increment DG ([11])

$$\bar{\nabla}_1 I(x, x') := \begin{pmatrix} \frac{I(x'_1, x_2, x_3, \dots, x_n) - I(x_1, x_2, x_3, \dots, x_n)}{x'_1 - x_1} \\ \frac{I(x'_1, x'_2, x_3, \dots, x_n) - I(x'_1, x_2, x_3, \dots, x_n)}{x'_2 - x_2} \\ \vdots \\ \frac{I(x'_1, x'_2, x'_3, \dots, x'_n) - I(x'_1, x'_2, \dots, x'_{n-1}, x_n)}{x'_n - x_n} \end{pmatrix}.$$

³The system (2) is Poisson if the skew matrix $S(x)$ satisfies the Jacobi identity:

$$\sum_{l=1}^n \left\{ S_{il} \frac{\partial}{\partial x_l} S_{jk} + S_{jl} \frac{\partial}{\partial x_l} S_{ki} + S_{kl} \frac{\partial}{\partial x_l} S_{ij} \right\} = 0 \text{ for } i, j, k = 1, \dots, n.$$

⁴The exceptions here are for quantities that are defined in terms of derivatives, viz. B_{ij} , M_{ijk} , \mathcal{H}_{ij} , D_{im} , P_{ijm} .

(iv) Symmetrized version of (iii)

$$\bar{\nabla}_3 I(x, x') := \frac{1}{2} \left(\bar{\nabla}_1 I(x, x') + \bar{\nabla}_1 I(x', x) \right)$$

Series expansion for discrete gradients

The general discrete gradient $\bar{\nabla} I$ may be expanded in the form

$$\begin{aligned} \bar{\nabla} I(x, x') &= \nabla I(x) + B(x)(x' - x) \\ &+ (x' - x)^T \mathbf{M}(x)(x' - x) \\ &+ O(\|x' - x\|^3). \end{aligned} \quad (4)$$

From (3) (definition of DG) it follows that

$$B_{ij} + B_{ji} = I_{ij} \text{ and } M_{ijk} + M_{jki} + M_{kij} = \frac{1}{2} I_{ijk} \quad (5)$$

For symmetric DG's (for which $\bar{\nabla} I(x, x') = \bar{\nabla} I(x', x)$, e.g. $\bar{\nabla}_3 I, \bar{\nabla}_{MV} I, \bar{\nabla}_{MP} I$),

$$B = \frac{1}{2} \mathcal{H}$$

where \mathcal{H} is the Hessian, $\mathcal{H}_{ij} := \frac{\partial^2 I}{\partial x_i \partial x_j} = I_{ij}$

Example: For $\bar{\nabla}_1 I$,

$$B_{ij} = \begin{cases} 0 & \text{if } i < j \\ \frac{1}{2} I_{ii} & \text{if } i = j \\ I_{ji} & \text{if } i > j \end{cases} \quad (6)$$

and for the tensor \mathbf{M} , each of the symmetric matrices M_k , $k = 1, \dots, n$ is defined by

$$M_{kij} = \begin{cases} 0 & i, j > k \text{ if } k \leq n-1 \\ \frac{1}{2} I_{kii} & i = 1, 2, \dots, k-1, j = i \\ \frac{1}{6} I_{iii} & \text{if } k = j = i \\ \frac{1}{2} I_{kij} & j = 2, 3, \dots, k-1, i = 1, 2, \dots, j-1 \\ \frac{1}{4} I_{kik} & i = 1, 2, \dots, k-1, j = k \\ M_{kji} & \text{(symmetric), } j > i \end{cases} \quad (7)$$

Conditions (5) are satisfied by (6) and (7) respectively.

4 Discrete gradient integrators

An integral-preserving discrete version of equation (2) is

$$\frac{(x' - x)}{\tau} = \tilde{S}(x, x', \tau) \bar{\nabla} I(x, x') \quad (8)$$

where x, x' denote x_n resp. x_{n+1} , and where \tilde{S} is a skew-symmetric matrix satisfying (for consistency)

$$\tilde{S}(x, x', \tau) = S(x) + O(\tau).$$

The integrator (8) is usually first order if $\tilde{S} = S$. Furthermore, (8) is symmetric (self-adjoint) and hence second order if both $\tilde{S}(x, x', \tau) = \tilde{S}(x', x, -\tau)$ and $\bar{\nabla}I(x, x') = \bar{\nabla}I(x', x)$. For example, the former condition is automatically satisfied if S is a constant matrix; if it is not constant, one can use

$$\tilde{S}(x, x', \tau) = S\left(\frac{x + x'}{2}\right)$$

and use any symmetric DG for $\bar{\nabla}I(x, x')$.

5 Bootstrapping to higher order

We now show how the order of a discrete gradient IPI can be bootstrapped to higher values. We first demonstrate the method with a bootstrap from first to second order, deriving a correction term that is itself integral-preserving. This is followed by a proof that the correction term required to bootstrap from n^{th} order to $(n + 1)^{\text{st}}$ order is always integral-preserving. We then proceed to two different bootstraps from second to third order, and a discussion of ways to construct a fourth order IPI.

From first order to second order

Starting with the first-order integrator

$$\frac{(x' - x)}{\tau} = S_1 \bar{\nabla}I(x, x'), \quad (9)$$

where $S_1 := S = S(x)$, substitution of $(x' - x)$ from (9) into the second term of (4) gives the approximation

$$\bar{\nabla}I(x, x') = (Id_n + \tau BS)\nabla I(x) + O(\tau^2), \quad (10)$$

where Id_n is the $n \times n$ identity matrix.

Differentiation of (1) gives

$$\ddot{x} = (C + S\mathcal{H}S)\nabla I, \text{ where } C^{ij} = S_m^{ij} S^{ml} I_l. \quad (11)$$

Then

$$(x' - x)_{\text{exact}} - (x' - x)_{1^{\text{st}} \text{ order IPI}} = \tau^2 R_2 \nabla I + O(\tau^3), \quad (12)$$

where $R_2 = SQS + \frac{1}{2}C$, with skew matrix $Q(x) := \frac{1}{2}\mathcal{H}(x) - B(x)$. Both SQS and C are skew, so $(\nabla I)^T R_2 \nabla I = 0$. The RHS of (12) provides an integral-preserving correction term $R_2 \nabla I$ that leads to the second-order integrator

$$\frac{(x' - x)}{\tau} = S_2(x) \bar{\nabla}I(x, x'). \quad (13)$$

Here,

$$S_2(x) := S + \tau R_2 = S + \tau(SQS + \frac{1}{2}C). \quad (14)$$

Bootstrap from n^{th} order to $(n + 1)^{\text{st}}$ order

Backward error analysis can be used to show that the correction term at each increase in order is integral-preserving, assuming an integral-preserving modified vector field \tilde{f} :

The exact flow is given by $\phi_t = e^{t f(x) \frac{\partial}{\partial x}}$, and a single step of the n^{th} order IPI flow ψ_τ by

$$\begin{aligned} \psi_\tau &= e^{\tau \tilde{f}(x) \frac{\partial}{\partial x}} \\ &= e^{(\tau f(x) \frac{\partial}{\partial x} + \tau^{n+1} \tilde{f}(x) \frac{\partial}{\partial x})} \\ &= e^{\tau f(x) \frac{\partial}{\partial x}} e^{\tau^{n+1} \tilde{f}(x) \frac{\partial}{\partial x}} + O(\tau^{n+2}) \\ &= e^{\tau f(x) \frac{\partial}{\partial x}} \cdot (1 + \tau^{n+1} \tilde{f}(x) \frac{\partial}{\partial x}) + O(\tau^{n+2}) \\ &= (e^{\tau f(x) \frac{\partial}{\partial x}} + \tau^{n+1} \tilde{f}(x) \frac{\partial}{\partial x}) + O(\tau^{n+2}) \end{aligned}$$

Hence

$$\begin{aligned} (x' - x)_{\text{exact}} - (x' - x)_{n^{\text{th}} \text{ order IPI}} &= \tau^{n+1} \tilde{f}(x) + O(\tau^{n+2}) \\ &= \tau^{n+1} \tilde{S}(x) \nabla I + O(\tau^{n+2}) \end{aligned}$$

where \tilde{S} is skew since \tilde{f} is integral-preserving.

I.e. we can write

$$(x' - x)_{\text{exact}} - (x' - x)_{n^{\text{th}} \text{ order IPI}} = \tau^{n+1} R_{n+1}(x) \nabla I + O(\tau^{n+2})$$

with skew R_{n+1} , i.e.

$$(x' - x)_{\text{exact}} - (\tau \tilde{S}_n \bar{\nabla} I + \tau^{n+1} R_{n+1} \bar{\nabla} I) = O(\tau^{n+2}),$$

and the $(n+1)^{\text{st}}$ order IPI is

$$\frac{x' - x}{\tau} = \tilde{S}_{n+1}(x) \bar{\nabla} I + O(\tau^{n+2}).$$

Here, the skew matrix \tilde{S}_{n+1} is defined by

$$\tilde{S}_{n+1}(x) := \tilde{S}_n(x) + \tau^n R_{n+1}(x).$$

From second order to third order

The second-order integral-preserving integrator (13) can thus be bootstrapped to third order. Differentiation of (11) gives

$$\ddot{x} = (A + N + K + 2CHS + SHC + SDS + SHSHS) \nabla I,$$

where

$$\begin{aligned} A^{im} &= S_{kj}^{im} S^{kl} I_l S^{jn} I_n \\ N^{im} &= S_j^{im} S_k^{jn} S^{kl} I_l I_n \\ K^{im} &= S_j^{im} S^{jn} I_{np} S^{pk} I_k \\ D_{im} &= I_{imn} S^{nk} I_k \end{aligned}$$

We then obtain

$$(x' - x)_{\text{exact}} - (x' - x)_{2^{\text{nd}} \text{ order IPI}} = \tau^3 R_3 \bar{\nabla} I + O(\tau^4) \quad (15)$$

with

$$\begin{aligned} R_3 := & \quad S Q S Q S + \frac{1}{12} (C H S - S H C) - \frac{1}{12} S H S H S \\ & + \frac{1}{2} (S Q C + C Q S) + \frac{1}{6} (A + N + K) + V \end{aligned} \quad (16)$$

where

$$V^{kn} = \frac{2}{3} S^{ki} (P_{ijm} - P_{jim}) S^{mp} I_p S^{jn} \quad (17)$$

and

$$P_{ijm} := \frac{1}{6} I_{ijm} - M_{ijm}. \quad (18)$$

From (5) it can be shown that

$$P_{ijm} + P_{jmi} + P_{mij} = 0. \quad (19)$$

All the terms $S Q S Q S$, $C H S - S H C$, $S H S H S$, $S Q C + C Q S$, A , N , K and V are skew-symmetric matrices, hence R_3 is also skew, as expected.

We thus obtain a third-order IPI as follows:

$$\frac{x' - x}{\tau} = S_3(x) \bar{\nabla} I(x, x'), \quad (20)$$

with

$$S_3 := S_2 + \tau^2 R_3 = S + \tau (S Q S + \frac{1}{2} C) + \tau^2 R_3. \quad (21)$$

When S is constant (e.g. for a standard Hamiltonian system), this simplifies to

$$S_3 = S + \tau S Q S + \tau^2 (S Q S Q S - \frac{1}{12} S H S H S + V).$$

A possible alternative bootstrap from second order to third order

The bootstrap process carried out above did not specify the use of any particular discrete gradient. We now describe a bootstrap that starts from an IPI that is second-order because it is self-adjoint (symmetric). This is the case for the integrator

$$\frac{(x' - x)}{\tau} = \tilde{S}(x, x', \tau) \bar{\nabla} I(x, x')$$

if $\tilde{S}(x, x', \tau) = \tilde{S}(x', x, -\tau)$, and $\bar{\nabla} I(x, x')$ is symmetric, i.e. $\bar{\nabla} I(x, x') = \bar{\nabla} I(x', x)$. For example, these requirements are satisfied if we use

$$\tilde{S}(x, x', \tau) = S \left(\frac{x + x'}{2} \right) \text{ and } \bar{\nabla} I(x, x') = \bar{\nabla}_3 I(x, x').$$

The third-order integrator in this case is

$$\frac{x' - x}{\tau} = S'_3(x, x') \bar{\nabla} I(x, x'), \quad (22)$$

with

$$S'_3 := S \left(\frac{x+x'}{2} \right) + \tau^2 R'_3 \quad (23)$$

and

$$R'_3(x) := \frac{1}{12}(CHS - SHC - SHSHS) + \frac{1}{24}(A - 2N - 2K) + V. \quad (24)$$

When S is constant, this simplifies to

$$R'_3 = -\frac{1}{12}SHSHS + V.$$

From third order to fourth order

Fourth order IPI's can be gained in 3 ways:

(i) A symmetric 4th order integrator results if instead of $R'_3(x)$, $R'_3\left(\frac{x+x'}{2}\right)$ is used in the integrator (22), (later referred to as SA4). In this case the integrator is

$$\frac{x' - x}{\tau} = S'_3 \left(\frac{x+x'}{2} \right) \bar{\nabla} I(x, x')$$

where $\bar{\nabla} I(x, x')$ is any symmetric discrete gradient,

$$S'_3 := S \left(\frac{x+x'}{2} \right) + \tau^2 R'_3 \left(\frac{x+x'}{2} \right),$$

$$R'_3(x) := \frac{1}{12}(CHS - SHC - SHSHS) + \frac{1}{24}(A - 2N - 2K) + V,$$

and A , N and K are given by

$$\begin{aligned} A^{im} &= S_{kj}^{im} S^{kl} I_l S^{jn} I_n \\ N^{im} &= S_j^{im} S_k^{jn} S^{kl} I_l I_n \\ K^{im} &= S_j^{im} S^{jn} I_{np} S^{pk} I_k \end{aligned}$$

and V by equation (17).

(ii) If ϕ_τ is either of the 3rd order integrators

$$\frac{x' - x}{\tau} = S_3(x) \bar{\nabla} I(x, x'), \text{ or } \frac{x' - x}{\tau} = S'_3(x) \bar{\nabla} I(x, x')$$

then the composition $\phi_{\frac{\tau}{2}} \circ \phi_{\frac{\tau}{2}}^{-1}$ will be 4th order (for example this is the method referred to as DM in [21]).

(iii) Use a composition method based on symmetric 2nd order IPI

$$\frac{x' - x}{\tau} = S \left(\frac{x+x'}{2} \right) \bar{\nabla} I(x, x') \quad (25)$$

with a symmetric DG, (e.g. $\bar{\nabla}_3 I(x, x')$) to make the 4th order IPI ψ_τ . In seeking an optimal composition (i.e. one that is fast and accurate) we have tried methods due to Yoshida [32], Suzuki [30] and McLachlan [15]. The latter method is a 10-stage composition based on the 1st order IPI (9).

Another alternative would be to perform one more bootstrap, a fairly onerous task.

6 Numerical experiments

To demonstrate the possible advantages of these new integrators we will present a comparison of the Self-adjoint Discrete Mechanics fourth-order integrator SA4, obtained from (22) and summarised above, with two other symmetric 4th order integrators. We will also consider the 2nd order integrator (25) and the 3rd order integrator (22), in order to better understand the circumstances in which our method is preferable.

6.1 The differential equations

The three systems of differential equations considered are described below. They will be referred to as FPU, modified FPU, and QT.

• FPU is the (Hamiltonian) Fermi-Pasta-Ulam system (4 springs, 3 particles, fixed ends [31]). The differential equations are

$$\begin{aligned}
 \dot{x}_1 &= x_4 \\
 \dot{x}_2 &= x_5 \\
 \dot{x}_3 &= x_6 \\
 \dot{x}_4 &= -2x_1 + x_2 + \alpha(x_2^2 - 2x_1x_2) \\
 \dot{x}_5 &= x_1 - 2x_2 + x_3 + \alpha(-x_1^2 + x_3^2 + 2x_1x_2 - 2x_2x_3) \\
 \dot{x}_6 &= x_2 - 2x_3 + \alpha(-x_2^2 + 2x_2x_3)
 \end{aligned} \tag{26}$$

and the Hamiltonian is

$$\begin{aligned}
 H(x) &= \frac{1}{2}(x_1^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2 + (x_2 - x_1)^2 + (x_3 - x_2)^2) \\
 &\quad + \frac{\alpha}{3}(x_1^3 - x_3^3 + (x_2 - x_1)^3 + (x_3 - x_2)^3).
 \end{aligned} \tag{27}$$

The skew matrix $S(x)$ for this system is

$$S(x) = \begin{pmatrix} 0 & Id_3 \\ -Id_3 & 0 \end{pmatrix}. \tag{28}$$

where Id_3 is the 3×3 identity matrix. We used $\alpha = \frac{1}{4}$, and the initial conditions (corresponding to being initially at rest with all energy in the lowest frequency mode) were $x_1 = x_3 = 0.5, x_2 = \frac{1}{\sqrt{2}}, x_4 = x_5 = x_6 = 0$. The skew third-order correction matrix $C = R'_3(x)$ for this FPU system is given by

$$6C_{ij} = \begin{cases} 0 & i = 1, j = 2, 3, 6; i = 2, j = 3 \\ 0 & i = 3, j = 4; i = 4, j = 6 \\ 1 + \alpha x_2 & i = 1, j = 4 \\ -\frac{1}{2} + \alpha(x_1 - x_2) & i = 1, j = 5 \text{ and } i = 2, j = 4 \\ 1 - \alpha(x_1 - x_3) & i = 2, j = 5 \\ -\frac{1}{2} + \alpha(x_2 - x_3) & i = 2, j = 6 \text{ and } i = 3, j = 5 \\ 1 - \alpha x_2 & i = 3, j = 6 \\ -\alpha(x_4 + x_5) & i = 4, j = 5 \\ -\alpha(x_5 + x_6) & i = 5, j = 6 \end{cases} \tag{29}$$

• modified FPU is a perturbed version of the FPU system, with the same first integral as for FPU, and skew matrix

$$S(x) = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 + x_m^n \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 - x_m^n & 0 & 0 & 0 \end{pmatrix}. \quad (30)$$

Variation of m and n will change the dynamics and the details of the third-order correction matrix. In particular, as n increases so does the complexity of the correction matrix and the consequent computational overhead of SA4. The selected examples used in the plots shown were for the cases $m = 3, n = 1$ (Figures 6.1(b) and 6.2(b)), and $m = 1, n = 4$ (Figures 6.1(c) and 6.2(c)). The former system is Poisson, the latter is not. An example of the changed correction matrix elements for these cases is the element C_{35} , which for FPU was given by

$$6C_{35} = -\frac{1}{2} + \alpha(x_2 - x_3).$$

For for $m = 3, n = 1$, it is

$$\begin{aligned} 6C_{35} &= -\frac{1}{2} + \alpha(x_2 - x_3) \\ &\quad + 2\alpha x_2 x_3 - 2\alpha x_3^2 - x_3 + \alpha x_2 x_3^2 - \alpha x_3^3 - \frac{1}{2} x_3^2. \end{aligned}$$

For the case $m = 1, n = 4$ it becomes

$$\begin{aligned} 6C_{35} &= -\frac{1}{2} + \alpha(x_2 - x_3) \\ &\quad + 2\alpha x_2 x_1^4 - 2\alpha x_3 x_1^4 - x_1^4 + \alpha x_2 x_1^8 - \alpha x_3 x_1^8 - \frac{1}{2} x_1^8. \end{aligned}$$

The same initial conditions as for FPU were used for modified FPU.

• QT is a 3-dimensional system devised by Quispel and Turner [26], with differential equations

$$\begin{aligned} \dot{x}_1 &= x_1 + x_2 x_3 + \frac{1}{10} x_1 x_2^2 + x_1 x_3^2 + \frac{1}{10} x_1 x_2^5 \\ \dot{x}_2 &= -x_1^2 - x_3^2 - \frac{1}{10} x_1^2 x_2^2 \\ \dot{x}_3 &= -x_3 - x_1 x_2 - x_2^3 x_3 \end{aligned} \quad (31)$$

and the first integral is

$$I(x) = \frac{1}{2}(x_1^2 + x_3^2) + \frac{1}{4}x_2^2 + x_2. \quad (32)$$

The skew matrix $S(x)$ for this system is

$$S(x) = \begin{pmatrix} 0 & x_1 + \frac{1}{10}x_1x_2^2 & x_2 \\ -x_1 - \frac{1}{10}x_1x_2^2 & 0 & x_3 \\ -x_2 & -x_3 & 0 \end{pmatrix}. \quad (33)$$

The terms in the skew third-order correction matrix $C = R'_3(x)$ are lengthy high-order polynomials in the 3 variables.

The initial conditions were $x_1 = x_3 = 1$, $x_2 = 2$.

6.2 Comparisons of different 4th order methods

The three methods used were:

- SA4, the 4th order self-adjoint bootstrapped integrator described above,
- SU4, a 5-stage composition method due to Suzuki [30]⁵, and
- SP4, a symmetric projection method due to Hairer, the details of which are given in an Appendix.

Regarding the choice of composition method, it was found that for the FPU and modified FPU systems the Suzuki [30] and McLachlan [15] methods gave almost collinear results in the error vs CPU time plots (with McLachlan more accurate for a given time-step). In the case of the QT system, Suzuki was a little more efficient, and so SU4 was chosen. Both McLachlan and Suzuki were more efficient than the Yoshida composition [32] in all the cases considered here.

All numerical computations were executed using double precision Fortran on a Dual Processor 2 GHz Macintosh G5. Maple[®] programs were used to help produce the Fortran code for the bootstrap correction terms and for the symmetric projections.

For the data in Figures 1(a), 1(b) and 1(c), the initial step-size was $\tau = 0.1$, reduced by a factor of 1.1 before each repeat of the computations (done 24 times), with $t_{max} = 10^4$. In Figure 1(d) the initial step-size was $\tau = 0.01$, and $t_{max} = 10^4$.

In all the cases shown, SP4 is slower than the other two methods. Inspection of Figures 1(a), 1(b) and 1(c) indicates that the SA4 results move steadily from being clearly the best (for FPU, Fig 1(a)) to being no better than SU4 for the quartically perturbed modified FPU system of Fig 1(c). Further computations not shown here indicate that this trend does not depend on whether the modified FPU system used is Poisson or not.

The combination of quadratic integral and lower dimension cubic $S(x)$ -matrix in the QT system leads to computationally very expensive correction terms. The resulting relatively poor performance of SA4 for this system is shown in Figure 1(d).

A least-squares fit to the global error (E) vs step-size (τ) data for the three 4th-order integrators SA4, SU4 and SP4 yields (for FPU), respectively,

$$E_{SA4} \approx 71.42\tau^{3.999}, \quad E_{SU4} \approx 7.972\tau^{3.999} \text{ and } E_{SP4} \approx 13.17\tau^{3.999} \quad (\text{FPU})$$

⁵The second-order integrator used to obtain SU4 was that of equation (25)

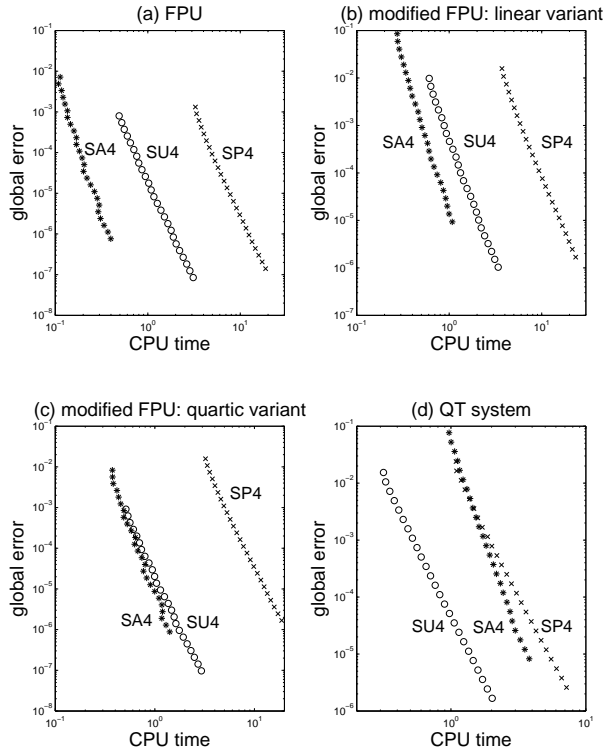


Figure 1: Global error vs CPU time for the three 4th order integrators SA4, SU4 and SP4, applied to (a) FPU system, (b) and (c) variants of the FPU system, and (d) the QT system. All computations are done for 25 step sizes – starting at (for (a), (b) and (c)) $\tau_0 = 0.1$ and reducing exponentially by a factor of 1.1, with $t_{max} = 10^4$. For (d) the initial step size was $\tau_0 = 0.01$, and $t_{max} = 10^3$.

6.3 Conclusion for the 4th order methods

We can conclude that in these experiments SP4 is always worst, SA4 starts off better than SU4 for sparse and low order polynomial $S(x)$ -matrix, but is worse for dense high-order polynomial $S(x)$ -matrix.

6.4 The 2nd and 3rd order methods

Figure 2 shows the global error versus CPU time plots resulting from the application of the second-order symmetric integrator SA2, equation (25), and the bootstrapped third-order integrator SA3, equation (22). The initial step size for (a), (b) and (c) was 10^{-2} , with $t_{max} = 2 \times 10^3$. To achieve the intersections shown in case (d) different

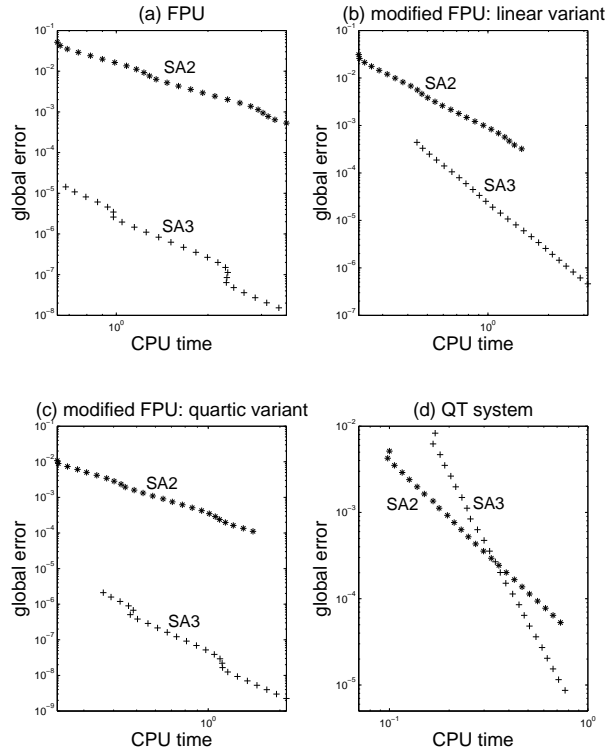


Figure 2: FPU system: Global error vs CPU time for the 2nd and 3rd order integrators SA2 and SA3, applied to (a) FPU system, (b) and (c) variants of the FPU system, and (d) the QT system. All computations are done for 25 step sizes – starting at (for (a), (b) and (c)) $\tau_0 = 0.01$ and reducing exponentially by a factor of 1.1, with $t_{max} = 2 \times 10^3$. For (d) the initial step size was $\tau_0 = 4 \times 10^{-4}$ (for SA2), $\tau_0 = 5 \times 10^{-3}$ (for SA3), and $t_{max} = 10^2$.

initial step-sizes were used: for SA2, $\tau = 4 \times 10^{-4}$, and for SA3, $\tau = 5 \times 10^{-3}$, with $t_{max} = 10^2$. The reduction factor was 1.1, , applied 24 times, as before. Note that SA4 is created by making SA3 symmetric, as explained earlier. Same-work computations comparing SA3 and SA4 when applied to the FPU system indicate that SA4 is better as it is more accurate.

The point of these results is that they give an alternative view of the extra computational load resulting from the correction terms which change the second order integrator SA2 into the third order integrator SA3. This is most clearly evident in the increased CPU time required, for a given step size, Figs 2(a – c). For the QT system, Fig 2(d), the extra load is so great that SA2 can be more efficient than SA3 (for acceptable error size).

Appendix: Symmetric projection

Hairer [6,7] has published a symmetric method for projection of a numerical integrator onto an invariant manifold. We employ in this paper an adaptation of Hairer's method using as the underlying symmetric integrator the 4th-order Gauss method.

Suppose that for the initial-value problem

$$x' = f(x), \quad x(0) = x_0$$

there is an invariant manifold \mathcal{M} defined by

$$\mathcal{M} = \{x \in \mathbb{R}^n; g(x) = 0\}, \text{ with } g'(x)f(x) = 0 \text{ for } x \in \mathcal{M}, \quad g: \mathbb{R}^n \mapsto \mathbb{R}^m.$$

At each step, assuming the initial point to be on the manifold, the method first applies a perturbation off the manifold, then a symmetric one-step integrator Φ_τ , then projection back onto the manifold. Both perturbation and projection are done so as to preserve the symmetry of the whole process. With orthogonal projection we have:

$$\begin{aligned} \hat{x}_p &= x_p + G^T(x_p)\mu \quad \text{with } g(x_p) = 0, \text{ (the perturbation),} \\ \hat{x}_{p+1} &= \Phi_\tau(\hat{x}_p) \\ x_{p+1} &= \hat{x}_{p+1} + G^T(x_{p+1})\mu \quad \text{with } g(x_{p+1}) = 0, \text{ (the projection).} \end{aligned}$$

Here, $G(x) = g'(x)$ is the Jacobian of g . The components of the vector $\mu \in \mathbb{R}^m$ are the Lagrange multipliers, and μ is defined implicitly by

$$F(\mu) = g(\Phi_\tau(\hat{x}_0 + G^T(x_0)\mu) + G^T(x_1)\mu) = 0$$

which can be solved using Newton's method.

If Φ_τ is the Gauss 4th-order integrator, at each time-step we must solve the following $(3n + m)$ -dimensional system of equations:

$$\begin{aligned} d &= x_{p+1} - G^T(x_{p+1})\mu - x_p - G^T(x_p)\mu - \tau(b_1 f(Y_1) + b_2 f(Y_2)) = 0 \\ e &= g(x_{p+1}) = 0 \\ v_1 &= Y_1 - x_p - G^T(x_p)\mu - \tau(a_{11} f(Y_1) + a_{12} f(Y_2)) = 0 \\ v_2 &= Y_2 - x_p - G^T(x_p)\mu - \tau(a_{21} f(Y_1) + a_{22} f(Y_2)) = 0 \end{aligned}$$

with $b_1 = b_2 = \frac{1}{2}$, $a_{11} = a_{22} = \frac{1}{4}$, $a_{12} = \frac{1}{4} - \frac{\sqrt{3}}{6}$, $a_{21} = \frac{1}{4} + \frac{\sqrt{3}}{6}$.

Newton's method for solving this system can be written as

$$\begin{pmatrix} \Delta Y_1 \\ \Delta Y_2 \\ \Delta x_{p+1} \\ \Delta \mu \end{pmatrix} = J^{-1} \cdot \begin{pmatrix} v_1 \\ v_2 \\ d \\ e \end{pmatrix}$$

where J is the Jacobian of $(d, e, v_1, v_2)^T$ with respect to $(x_{p+1}, \mu, Y_1, Y_2)^T$. J can be expanded as a series in powers of τ , $J = J_0 + O(\tau)$, where

$$J_0 = \begin{pmatrix} Id_n & 0 & 0 & -G^T(x_p) \\ 0 & Id_n & 0 & -G^T(x_p) \\ 0 & 0 & Id_n & -2G^T(x_p) \\ 0 & 0 & G(x_p) & 0 \end{pmatrix}.$$

Hairer [6] gives the corresponding “simplified” Jacobian for the case of a second order integrator Φ_τ .

If J_0 is used, the Newton iteration requires more iterations to converge than if the full Jacobian J is used, but it may be faster in terms of CPU time. We have tried both approaches, and have also experimented with controlling the number of times J is recalculated during the time-step iteration, and with different starting values for the Newton iteration. Our aim was to reduce the CPU time as much as possible. For the systems studied here, the fastest computations resulted from the use of a single evaluation of J^{-1} per 4 time-steps, combined with starting values predicted using the 2^{nd} order Laburta method [7, 12]. This combination ran a little faster than that using J_0 , and a lot faster than code allowing unfettered evaluations of J . Some further improvements in the speed may be possible, but they are unlikely to bridge the gap between SP and SA4. A higher order version of Laburta’s starter was not used, since adapting the 2^{nd} order starter to this purpose (symmetric projection rather than a straight Gauss integration) already requires one extra function evaluation.

Acknowledgement

We are grateful to the Australian Research Council for financial support.

References

- [1] U. Ascher and S. Reich, *On some difficulties in integrating highly oscillatory Hamiltonian systems*, in *Computational Molecular Dynamics*, Lect. Notes Comput. Sci. Eng. **4**, Springer (1999) pp. 281–296.
- [2] C.J. Budd and A. Iserles (eds.), *Geometric integration: numerical solution of differential equations on manifolds*, Phil. Trans. R. Soc. Lond. A **357** (1999) pp. 943–1133.
- [3] C.W. Gear, *Maintaining solution invariants in the numerical solution of ODEs*, SIAM J. Sci. Stat. Comput. **7** (1986) pp. 734–743.
- [4] O. Gonzalez, *Time integration and discrete Hamiltonian systems*, J.Nonlinear Sci. **6** (1996) pp. 449–467.
- [5] O. Gonzalez and J.C. Simo, *On the stability of symplectic and energy-momentum algorithms for nonlinear Hamiltonian systems with symmetry*, Computer Methods in Applied Mathematics and Engineering, **134** (1996) pp. 197–222
- [6] E. Hairer, *Symmetric projection methods for differential equations on manifolds*, BIT **40** (2000) pp. 726–734.
- [7] E. Hairer, C. Lubich and G. Wanner, *Geometric Numerical Integration*, Springer Series in Computational Mathematics **31**, Springer-Verlag 2002.

- [8] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*, Springer Series in Computational Mathematics **14** Springer-Verlag 1996.
- [9] A. Harten, P.D. Lax and B. van Leer, *On upstream differencing and Godunov-type schemes for hyperbolic conservation laws*, SIAM Rev **25** (1983) pp. 35–61.
- [10] A. Iserles, H.Z. Munthe-Kaas, S.P. Norsett and A. Zanna, *Lie-group methods*, Acta Numerica (2000) pp. 215–365.
- [11] T. Itoh and K. Abe, *Hamiltonian-Conserving Discrete Canonical Equations Based on Variational Difference Quotients*, J. Comput. Phys. **77** (1988) pp. 85–102
- [12] M.P. Laburta, *Construction of starting algorithms for the RK-Gauss methods*, J. Comput. Appl. Math. **90** (1998) pp. 239–261.
- [13] B. Leimkuhler and S. Reich, *Simulating Hamiltonian Dynamics*, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press 2004.
- [14] R.I. McLachlan, *Explicit Lie-Poisson integration and the Euler equations*, Phys. Rev. Lett. **71** (1993) pp. 3043–3046.
- [15] R.I. McLachlan, *On the numerical integration of ordinary differential equations by symmetric composition methods*, SIAM J. Sci. **16** (1995) pp. 151–168.
- [16] R.I. McLachlan, G.R.W. Quispel and N. Robidoux, *Geometric integration using discrete gradients*, Phil. Trans. Roy. Soc. A **357** (1999) pp. 1021–1045.
- [17] R.I. McLachlan and G.R.W. Quispel, *Six lectures on the geometric integration of ODEs*, in Foundations of Computational Mathematics, C.U.P. (2001), R.A. DeVore et al. (eds.), pp. 155–210.
- [18] R.I. McLachlan and G.R.W. Quispel, *Splitting methods*, Acta Numerica **11** (2002) pp. 341–434.
- [19] R.I. McLachlan and G.R.W. Quispel, *Geometric integrators for ODE's*, J. Phys. A: Math. Gen. **39**, Special issue on geometric integration of differential equations, (2006) pp. 5251–5285.
- [20] R.I. McLachlan, M. Perlmutter and G.R.W. Quispel, *Lie group foliations: dynamical systems and integrators*, Future Gen. Comp. Systems **19** (2003) pp. 1207–1219.
- [21] D.I. McLaren and G.R.W. Quispel, *Integral preserving integrators*, J. Phys. A: Math. Gen. **37** (2004) pp. L489–L495.
- [22] Qin Meng-Zhao and Zhu Wen-Jie, *Construction of higher order symplectic schemes by composition*, Computing **47** (1992) pp. 309–221.

- [23] G.R.W. Quispel, *Volume-preserving integrators*, Phys. Lett. A **206** (1995) pp. 26–30.
- [24] G.R.W. Quispel and H.W. Capel, *Solving ODEs numerically while preserving a first integral*, Phys. Lett. A **218** (1996) pp. 223–228.
- [25] G.R.W. Quispel and D.I. McLaren, *A new class of energy-preserving numerical integration methods*, J. Phys. A: Math. Theor. **41** (2008).
- [26] G.R.W. Quispel and G.S. Turner, *Discrete gradient methods for solving ODEs numerically while preserving a first integral*, J. Phys. A **29** (1996) pp. L341–L349.
- [27] J.M. Sanz-Serna and M.P. Calvo, *Numerical Hamiltonian Problems*, Chapman & Hall, London, 1994.
- [28] L.F. Shampine, *Conservation laws and the numerical solution of ODEs II* Computers and mathematics with Applications **38** (1999) pp. 61–72.
- [29] Shang Zai-jiu, *Construction of volume-preserving difference schemes for source-free systems via generating functions*, J. Comp. Math. **12** (1994) pp. 265–272.
- [30] M. Suzuki, *Fractal decomposition of exponential operators with applications to many-body theories and Monte Carlo simulations*, Phys. Lett. A **146** (1993) pp. 319–323.
- [31] T.P. Weisert, *The Genesis of Simulation in Dynamics: Pursuing the Fermi-Pasta-Ulam Problem*, Springer-Verlag 1997.
- [32] H. Yoshida, *Construction of higher order symplectic integrators*, Phys. Lett. A **150** (1990) pp. 262–268.