

# Efficient implementation of symplectic implicit Runge-Kutta schemes with simplified Newton iterations

Mikel Antoñana, Joseba Makazaga, Ander Murua  
KZAA saila, Informatika Fakultatea, UPV/EHU  
Donostia / San Sebastián

## Abstract

We are concerned with the efficient implementation of symplectic implicit Runge-Kutta (IRK) methods applied to systems of (non-necessarily Hamiltonian) ordinary differential equations by means of Newton-like iterations. We pay particular attention to symmetric symplectic IRK schemes (such as collocation methods with Gaussian nodes). For a  $s$ -stage IRK scheme used to integrate a  $d$ -dimensional system of ordinary differential equations, the application of simplified versions of Newton iterations requires solving at each step several linear systems (one per iteration) with the same  $sd \times sd$  real coefficient matrix. We propose rewriting such  $sd$ -dimensional linear systems as an equivalent  $(s+1)d$ -dimensional systems that can be solved by performing the LU decompositions of  $[s/2] + 1$  real matrices of size  $d \times d$ . We present a C implementation (based on Newton-like iterations) of Runge-Kutta collocation methods with Gaussian nodes that make use of such a rewriting of the linear system and that takes special care in reducing the effect of round-off errors. We report some numerical experiments that demonstrate the reduced round-off error propagation of our implementation.

## 1 Introduction

The main goal of the present work is the efficient implementation of symplectic implicit Runge-Kutta schemes for stiff ODE (non-necessarily Hamiltonian) problems. Our primary interest is on geometric numerical integration, which motivates us to solve the implicit equations determining each step to full machine precision. The stiff character of the target problems leads us to solving the implicit equations by some modified version of Newton method. This typically requires repeatedly solving linear systems of equations with coefficient matrix of the form

$$(I_s \otimes I_d - h A \otimes J) \in \mathbb{R}^{sd \times sd} \quad (1)$$

where  $A \in \mathbb{R}^{s \times s}$  is the coefficient matrix of the RK scheme, and  $J$  is some common approximation of the Jacobian matrices evaluated at the stage values.

An standard approach, independently introduced in [12], [5], and [3], to efficiently solve such linear systems takes advantage of the special structure of the matrix (1). More specifically, (1) is similar to a block-diagonal matrix with  $s$  blocks  $I_d - h \lambda_j J \in \mathbb{R}^{d \times d}$  ( $j = 1, \dots, s$ ), one per eigenvalue  $\lambda_j$  of  $A$ . Typically, the coefficient matrix  $A$  of standard high order implicit RK schemes have  $[s/2]$  complex conjugate pairs of eigenvalues (plus a real one for odd  $s$ ).

The main contribution of the present paper is a technique for transforming  $s d$ -dimensional systems with coefficient matrix (1) into an equivalent  $(s + 1)d$ -dimensional systems that can be solved by performing the LU decomposition of  $[s/2] + 1$  real matrices of size  $d \times d$  (and some additional multiplication of matrices of the same size). We pay particular attention to implicit Runge-Kutta schemes that are both symmetric and symplectic. However, our technique is also applicable for non-symmetric symplectic IRK schemes, and also for some symmetric non-symplectic IRK schemes (see last paragraph in Subsection 3.3 for more details).

In addition, we present an algorithm that implements symmetric symplectic IRK schemes (such as RK collocation methods with Gaussian nodes) by making use of the above technique. Special care is taken to try to reduce the effect of round-off errors by adapting some techniques used (for the implementation of symplectic IRK schemes with fixed-point iterations) in [1].

The plan of the paper is as follows: Section 2 summarizes some standard material about implicit Runge-Kutta methods and Newton-like iterations and fix some notation. Section 3 presents our new technique to solve the simplified linear system of Newton iterations for symplectic IRK schemes. Section 4 is devoted to describe our implementation of symplectic IRK methods with Newton-like iterations. Some numerical results are reported in Section 5. A few concluding remarks can be found in Section 6.

## 2 Implementation of implicit Runge-Kutta schemes with Newton-like iterations

### 2.1 Implicit Runge-Kutta schemes

We consider initial value problems of systems of ODEs of the form

$$\frac{d}{dt}y = f(t, y), \quad y(t_0) = y_0, \quad (2)$$

where  $f : \mathbb{R}^{d+1} \rightarrow \mathbb{R}^d$  is a sufficiently smooth map and  $y_0 \in \mathbb{R}^d$ .

Given a time discretization  $t_0 < t_1 < t_2 < \dots$ , the numerical approximations  $y_n \approx y(t_n)$ , ( $n = 1, 2, \dots$ ) to the solution  $y(t)$  of the initial value problem (2) is obtained by means of a one-step integrator as

$$y_{n+1} = \Phi(y_n, t_n, t_{n+1} - t_n), \quad n = 0, 1, 2, \dots, \quad (3)$$

for a map  $\Phi : \mathbb{R}^{d+2} \rightarrow \mathbb{R}^d$  determined in some way from  $f : \mathbb{R}^{d+1} \rightarrow \mathbb{R}^d$ .

In the case of a  $s$ -stage implicit Runge-Kutta method, the map  $\Phi$  is determined in terms of the real coefficients  $a_{ij}$  ( $1 \leq i, j \leq s$ ) and  $b_i, c_i$  ( $1 \leq i \leq s$ ) as

$$\Phi(y, t, h) := y + h \sum_{i=1}^s b_i f(t + c_i h, Y_i), \quad (4)$$

where the *stage vectors*  $Y_i$  are implicitly defined as functions of  $(y, t, h) \in \mathbb{R}^{d+2}$  by

$$Y_i = y + h \sum_{j=1}^s a_{ij} f(t + c_j h, Y_j), \quad i = 1, \dots, s. \quad (5)$$

Typically,

$$c_i = \sum_{j=1}^s a_{ij}, \quad i = 1, \dots, s.$$

The equations (5) can be solved for the stage vectors  $Y_i$  by means of some iterative procedure, starting, for instance, from  $Y_i^{[0]} = y$ ,  $i = 1, 2, \dots, s$ . (In the non-stiff case, it is usually more efficient initializing the stage vectors with some other procedure that uses the stage values of the previous steps [6]).

A very simple iterative procedure is fixed point iteration. For stiff problems, fixed-point iteration is not appropriate, and Newton iteration may be used to compute the stage vectors  $Y_i$  from (5). For non-stiff problems, Newton iteration may still be an attractive option in some cases, in particular for very high precision computations (for quadruple precision or in arbitrary precision arithmetic calculations) if implemented with mixed-precision strategies [2] (which reduce the cost of the linear algebra and the evaluation of the Jacobians, performed in lower precision arithmetic than the evaluations of the right-hand side of the system of ODEs).

In any case, since at each Newton iteration  $s$  evaluations of the Jacobian matrix  $\frac{\partial f}{\partial y}$  and a LU decomposition of a  $sd \times sd$  matrix are required, some computationally cheaper variants are often used instead.

## 2.2 Newton-like iterations

Recall that a Newton iteration can be used to compute for  $k = 1, 2, \dots$  the approximations  $Y_i^{[k]}$  of  $Y_i$  ( $i = 1, \dots, s$ ) in (5) as follows:

$$1) \quad r_i^{[k]} := -Y_i^{[k-1]} + y + h \sum_{j=1}^s a_{ij} f(t + c_j h, Y_j^{[k-1]}), \quad i = 1, \dots, s, \quad (6)$$

2) Solve  $\Delta Y_i^{[k]}$  from

$$\Delta Y_i^{[k]} - h \sum_{j=1}^s a_{ij} J_j^{[k]} \Delta Y_j^{[k]} = r_i^{[k]} \quad i = 1, \dots, s, \quad (7)$$

where  $J_i^{[k]} = \frac{\partial f}{\partial y}(t + c_i h, Y_i^{[k]})$  for  $i = 1, \dots, s$ ,

$$3) \quad Y_i^{[k]} := Y_i^{[k-1]} + \Delta Y_i^{[k]}, \quad i = 1, \dots, s, \quad (8)$$

Observe that,  $s$  evaluations of the Jacobian matrix  $\frac{\partial f}{\partial y}$  and a LU decomposition of a  $sd \times sd$  matrix are required (in addition to  $s$  evaluations of  $f$ ) at each iteration. This is typically computationally too expensive, and some variants of the full Newton algorithm are implemented instead. Among others, the following alternatives are possible:

- Application of simplified Newton iterations. This consists on replacing the Jacobian matrices  $J_i^{[k]}$  in (7) by  $J_i^{[0]} = \frac{\partial f}{\partial y}(t + c_i h, Y_i^{[0]})$ . In that case, LU decomposition is done only once and the linear system

$$\Delta Y_i^{[k]} - h \sum_{j=1}^s a_{ij} J_j^{[0]} \Delta Y_j^{[k]} = r_i^{[k]} \quad i = 1, \dots, s, \quad (9)$$

has to be solved at each of the simplified Newton iterations. If the simple initialization  $Y_i^{[0]} = y$  ( $i = 1, \dots, s$ ) is considered (this is typically the case when solving stiff systems) and  $f$  does not depend on  $t$ , then  $J_i^{[0]} = J := \frac{\partial f}{\partial y}(y)$  for each  $i = 1, \dots, s$ , and the linear system (9) reduces to

$$(I_s \otimes I_d - h A \otimes J) \Delta Y^{[k]} = r^{[k]}, \quad (10)$$

where

$$Y^{[k]} = \begin{pmatrix} Y_1^{[k]} \\ \vdots \\ Y_s^{[k]} \end{pmatrix} \in \mathbb{R}^{sd}, \quad r^{[k]} = \begin{pmatrix} r_1^{[k]} \\ \vdots \\ r_s^{[k]} \end{pmatrix} \in \mathbb{R}^{sd}.$$

Even in the case where some initialization procedure other than  $Y_i^{[0]} = y$  is used, in practice the linear system (9) is often replaced by (10), where  $J$  is some common approximation of  $\frac{\partial f}{\partial y}(t + c_i h, Y_i^{[0]})$ ,  $i = 1, \dots, s$ . An appropriate choice [16] is  $J := \frac{\partial f}{\partial y}(t + \bar{c} h, \bar{y})$ , where  $\bar{c} = \frac{1}{s} \sum_{i=1}^s c_i$  (which

for symmetric methods gives  $\bar{c} = \frac{1}{2}$ ) and  $\bar{y} = \frac{1}{s} \sum_{i=1}^s Y_i^{[0]}$ . Often, it will be sufficient to evaluate instead of  $\frac{\partial f}{\partial y}$  a computationally cheaper approximation of it.

- Applying the original Newton iteration by solving the linear systems (7) with some iterative method [14] preconditioned by the inverse of the matrix

$$I_s \otimes I_d - h A \otimes J. \quad (11)$$

In practice, the linear systems (7) are only approximately solved with the iterative method. In such case, the resulting scheme is sometimes referred to as inexact Newton iteration [14]. Further variants of Newton-like iterations will be obtained if the Jacobian matrices are not updated at each iteration.

In any of the two alternatives above, one needs to repeatedly solve linear systems of the form

$$(I_s \otimes I_d - h A \otimes J) \Delta Y = r, \quad (12)$$

for given  $r \in \mathbb{R}^{sd}$ . From now on, we will refer to (12) as simplified linear system (of Newton-like iterations).

Of course, (12) may be solved by previously computing the LU decomposition of the full  $sd \times sd$  matrix (11), but this may be done more efficiently.

An standard approach [12][5][3] consists on diagonalizing the matrix  $A$  as  $\Lambda = S^{-1}AS = \text{diag}(\lambda_1, \dots, \lambda_s)$ , and computing the LU decomposition of the matrix

$$I_s \otimes I_d - h \Lambda \otimes J = (S^{-1} \otimes I_d) (I_s \otimes I_d - h A \otimes J) (S \otimes I_d).$$

In that case, one needs to compute the LU decomposition of a real (resp. complex)  $d \times d$  matrix for each distinct real eigenvalue (for each distinct pair of complex eigenvalues) of  $A$ .

Alternatively, some authors [4][10] propose solving (12) by an iterative procedure preconditioned by the inverse of

$$I_s \otimes I_d - h \tilde{A} \otimes J, \quad (13)$$

where  $\tilde{A} \in \mathbb{R}^{s \times s}$  is a matrix chosen so that the LU decomposition of (13) can be more efficiently computed than that of (11).

In next section, we propose a new technique to efficiently solve simplified linear systems (12) of Newton iterations, provided that the IRK scheme is symplectic.

### 3 Efficient solution of simplified linear systems for symplectic IRK schemes

#### 3.1 Symplectic IRK schemes

In what follows, we consider symplectic IRK schemes, that is [15], IRK schemes whose coefficients satisfy

$$b_i a_{ij} + b_j a_{ji} - b_i b_j = 0, \quad 1 \leq i, j \leq s. \quad (14)$$

Condition (14) guarantees that the discrete flow resulting from the application of the IRK scheme to an autonomous Hamiltonian system is symplectic, with important favorable consequences in the long-term behavior of the numerical solution [6]. Condition (14) also implies that, when applied to an ODE system with a quadratic invariant, then it is also a conserved quantity for the numerical solution provided by the IRK scheme.

Nevertheless, our interest in condition (14) is of a completely different nature: We will see that such a condition allows to solve efficiently linear systems of the form (12) for a given  $d \times d$  real matrix  $J$  and a given  $r \in \mathbb{R}^{sd}$ . We will pay particular attention to symplectic IRK schemes that additionally satisfy (possibly after some reordering of the stage values  $Y_i$ ) the symmetry condition [6]

$$\begin{aligned} b_{s+1-i} &= b_i, & c_{s+1-i} &= 1 - c_i, & 1 \leq i \leq s, \\ b_j &= a_{s+1-i, s+1-j} + a_{i,j}, & 1 \leq i, j \leq s, \end{aligned} \quad (15)$$

In particular, the IRK schemes of collocation type with Gaussian nodes are both symplectic and symmetric.

#### 3.2 Alternative symplecticity and symmetry characterizations

The map  $\Phi$  determining the steps (3) of a IRK scheme can be alternatively written as

$$\Phi(y, t, h) := y + z,$$

where the stage vectors  $Y_i \in \mathbb{R}^d$  and the increment  $z \in \mathbb{R}^d$  are implicitly defined as functions of  $(y, t, h) \in \mathbb{R}^{d+2}$  by

$$Y_i = y + \frac{z}{2} + h \sum_{j=1}^s \bar{a}_{ij} f(t + c_j h, Y_j), \quad i = 1, \dots, s, \quad (16)$$

$$z = h \sum_{i=1}^s b_i f(t + c_i h, Y_i), \quad (17)$$

where

$$\bar{a}_{ij} = a_{ij} - \frac{b_j}{2}, \quad 1 \leq i, j \leq s. \quad (18)$$

Condition (14) may be equivalently characterized in terms of the matrix  $\bar{A} = (\bar{a}_{ij})_{i,j=1}^s$  and the diagonal matrix  $B$  with diagonal entries  $b_1, \dots, b_s$ . Indeed, (14) is equivalent to the requirement that the real  $s \times s$  matrix  $(B\bar{A})$  be antisymmetric.

As for the symmetry condition (15), it reads

$$\begin{aligned} b_{s+1-i} &= b_i, & \bar{c}_{s+1-i} &= -\bar{c}_i, & 1 \leq i \leq s, \\ \bar{a}_{s+1-i, s+1-j} &= -\bar{a}_{i,j}, & 1 \leq i, j \leq s, \end{aligned} \quad (19)$$

where  $\bar{c}_i = c_i - \frac{1}{2}$  for  $i = 1, \dots, s$ .

### 3.3 Efficient solution of the linear systems of the form (12)

From now on, we will only consider, without loss of generality<sup>1</sup>, symplectic IRK schemes with invertible  $B$ . Since  $B\bar{A}$  is antisymmetric, so is  $B^{\frac{1}{2}}\bar{A}B^{-\frac{1}{2}}$ , which implies that  $\bar{A}$  is diagonalizable with all eigenvalues in the imaginary axis. This is equivalent to the existence of a  $s \times s$  invertible matrix  $Q$  such that

$$Q^{-1}\bar{A}Q = \begin{pmatrix} 0 & D \\ -D^T & 0 \end{pmatrix} \quad (20)$$

where  $D$  is a real diagonal matrix (with non-negative diagonal entries) of size  $m \times (s - m)$ , where  $m = \lfloor (s + 1)/2 \rfloor$  (and  $s - m = \lceil s/2 \rceil$ ).

We will next show that (20) may be exploited to solve efficiently linear systems of the form (12). Consider the implicit equations (16)–(17). Application of simplified Newton iteration to such implicit equations leads to linear systems of the form

$$\begin{aligned} (I_s \otimes I_d - h\bar{A} \otimes J) \Delta Y - \frac{1}{2}(e_s \otimes I_d) \Delta z &= r, \\ (-h e_s^T B \otimes J) \Delta Y + \Delta z &= 0, \end{aligned} \quad (21)$$

where  $e_s = (1, \dots, 1)^T \in \mathbb{R}^s$ . Clearly, if  $(\Delta Y, \Delta z)$  is solution of (21), then  $\Delta Y$  is solution of (12).

By virtue of (20), the linear system (21) is equivalent, with the change of variables  $\Delta Y = (Q \otimes I_d)W$  to

$$\begin{aligned} \begin{pmatrix} I_m \otimes I_d & -hD \otimes J \\ hD^T \otimes J & I_{s-m} \otimes I_d \end{pmatrix} W - \frac{1}{2}(Q^{-1}e_s \otimes I_d) \Delta z &= (Q^{-1} \otimes I_d) r, \\ -h(e_s^T B Q \otimes J) W + \Delta z &= 0, \end{aligned} \quad (22)$$

The blockwise sparsity pattern of the system (22) allows obtaining its LU decomposition by computing, in addition to several multiplications of matrices of size  $d \times d$ , the LU decompositions of  $\lceil s/2 \rceil + 1$  real matrices of size  $d \times d$ : the matrices

$$I_d + h^2 \sigma_i^2 J^2, \quad i = 1, \dots, \lceil s/2 \rceil,$$

<sup>1</sup>Any symplectic IRK method with  $b_i = 0$  for some  $i$  is equivalent to a symplectic IRK scheme with fewer stages and  $b_i \neq 0$  for all  $i$  [8]

where  $\sigma_1, \dots, \sigma_{\lfloor s/2 \rfloor} \geq 0$  are the diagonal entries in  $D$ , and an additional  $d \times d$  matrix obtained from the former. We will give more details in Subsection 3.4 in the case of symmetric symplectic IRK schemes.

It is worth remarking that such a technique for solving linear systems of the form (12) is not restricted to symplectic IRK schemes. It is enough that the corresponding matrix  $\bar{A}$  be diagonalizable with all its eigenvalues in the imaginary axis. This seems to be the case of several families of (non-symplectic) symmetric IRK methods of collocation type, in particular, for the nodes of Lobatto quadrature formulas, or if the nodes are either the zeros or the extrema of Chebyshev polynomials of the first kind.

### 3.4 The case of symmetric symplectic IRK schemes

In the present section, in addition to the symplecticity conditions, that guarantee that the matrix  $B^{\frac{1}{2}}\bar{A}B^{-\frac{1}{2}}$  is antisymmetric, we assume that the symmetry conditions (19) hold.

Consider the  $s \times s$  orthogonal matrix  $P = (P_1 \ P_2)$  such that, for  $x = (x_1, \dots, x_s)^T \in \mathbb{R}^s$ ,  $P_1^T x = (y_1 \cdots y_m)^T$ , and  $P_2^T x = (y_{m+1}, \dots, y_s)^T$ , where

$$\begin{aligned} y_i &= \frac{\sqrt{2}}{2}(x_{s+1-i} + x_i), \quad \text{for } i = 1, \dots, \lfloor s/2 \rfloor, \\ y_m &= x_m, \quad \text{if } s \text{ is odd,} \\ y_i &= \frac{\sqrt{2}}{2}(x_{s+1-i} - x_i), \quad \text{for } i = m+1, \dots, s, \end{aligned}$$

with  $m = \lfloor (s+1)/2 \rfloor$ .

The symmetry condition (19) implies that  $P_i^T B^{\frac{1}{2}}\bar{A}B^{-\frac{1}{2}}P_i = 0$  for  $i = 1, 2$ , and since by symplecticity  $B^{\frac{1}{2}}\bar{A}B^{-\frac{1}{2}}$  is an antisymmetric matrix, we conclude that the matrix  $\bar{A}$  is similar to

$$P^T B^{\frac{1}{2}}\bar{A}B^{-\frac{1}{2}}P = \begin{pmatrix} 0 & K \\ -K^T & 0 \end{pmatrix} \quad (23)$$

where  $K = P_1^T B^{\frac{1}{2}}\bar{A}B^{-\frac{1}{2}}P_2$  (which is a real matrix of size  $m \times (s-m) = \lfloor (s+1)/2 \rfloor \times \lfloor s/2 \rfloor$ ). Let  $K = UDV^T$  be the singular value decomposition of  $K$ , (where  $U \in \mathbb{R}^{m \times m}$  and  $V \in \mathbb{R}^{(s-m) \times (s-m)}$  are orthonormal matrices, and  $D \in \mathbb{R}^{m \times (s-m)}$  is a diagonal matrix with the singular values  $\sigma_1, \dots, \sigma_{s-m}$  of  $K$  as diagonal entries). We have that (20) holds with

$$Q = (Q_1 \ Q_2) = B^{-1/2}(P_1 \ P_2) \begin{pmatrix} U & 0 \\ 0 & V \end{pmatrix} = B^{-1/2}(P_1 U \ P_2 V),$$

and  $Q^{-1} = Q^T B$ . This implies that the linear system (21), with the change of variables

$$\Delta Y = (Q \otimes I_d)W = (Q_1 \otimes I_d)W' + (Q_2 \otimes I_d)W'' \quad (24)$$



is equivalent to (22). Due to the first symmetry conditions in (19),  $e_s^T B P_2 = 0$ , and hence  $e_s^T B Q_2 = e_s^T B P_2 V = 0$ , so that (22) reads

$$\begin{aligned} W' - h(D \otimes J) W'' - \frac{1}{2} (Q_1^T B e_s \otimes I_d) \Delta z &= (Q_1^T B \otimes I_d) r, \\ h(D^T \otimes J) W' + W'' &= (Q_2^T B \otimes I_d) r, \\ -h(e_s^T B Q_1 \otimes J) W' + \Delta z &= 0. \end{aligned}$$

By solving for  $W''$  from the second equation of the linear system above,

$$W'' = -h(D^T \otimes J) W' + (Q_2^T B \otimes I_d) r. \quad (25)$$

and substitution in the remaining two equations, one obtains

$$\begin{aligned} (I_m \otimes I_d + h^2 D D^T \otimes J^2) W' - \frac{1}{2} (Q_1^T B e_s \otimes I_d) \Delta z &= R, \\ -h(e_s^T B Q_1 \otimes J) W' + \Delta z &= 0. \end{aligned}$$

where  $R = (Q_1^T B \otimes I_d) r + h(D Q_2^T B \otimes J) r \in \mathbb{R}^{md}$ .

The linear system above can be rewritten in terms of

$$R = \begin{pmatrix} R_1 \\ \vdots \\ R_m \end{pmatrix}, \quad W' = \begin{pmatrix} W_1 \\ \vdots \\ W_m \end{pmatrix}$$

with  $R_i, W_i \in \mathbb{R}^d$ ,  $i = 1, \dots, m$ , as follows:

$$(I_d + h^2 \sigma_i^2 J^2) W_i - \frac{\alpha_i}{2} \Delta z = R_i, \quad i = 1, \dots, m, \quad (26)$$

$$-h J \sum_{i=1}^m \alpha_i W_i + \Delta z = 0. \quad (27)$$

where

$$\begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_m \end{pmatrix} = Q_1^T B e_s,$$

and  $\sigma_1 \geq \dots \geq \sigma_{\lfloor s/2 \rfloor}$  are the singular values of  $K$ , and if  $s$  is odd (in which case  $m = \lfloor (s+1)/2 \rfloor = \lfloor s/2 \rfloor + 1$ ), then  $\sigma_m = 0$ .

Thus, the unknown  $\Delta z \in \mathbb{R}^d$  can be obtained by solving the linear system

$$M \Delta z = h J \sum_{i=1}^m \alpha_i (I_d + h^2 \sigma_i^2 J^2)^{-1} R_i, \quad (28)$$

where

$$M = I_d + J \frac{h}{2} \sum_{i=1}^m \alpha_i^2 (I_d + h^2 \sigma_i^2 J^2)^{-1} \in \mathbb{R}^{d \times d}. \quad (29)$$

The unknowns in  $W' \in \mathbb{R}^{md}$  are then solved from (26), while  $W'' \in \mathbb{R}^{(s-m)d}$  may be obtained from (25).

The required solution  $\Delta Y$  of the original linear system (12), may finally be obtained from (24).

### 3.5 Alternative reformulation of symplectic IRK schemes

If the coefficients  $b_i, a_{ij}$  determining a symplectic IRK are replaced by floating point numbers  $\tilde{b}_i, \tilde{a}_{ij}$  that approximate them, then the resulting IRK scheme typically fails to satisfy the symplecticity conditions (14). This results [7] in a method that exhibits a linear drift in the value of quadratic invariants of the system and in the Hamiltonian function when applied to autonomous Hamiltonian systems.

Motivated by that, the map  $\Phi : \mathbb{R}^{d+2} \rightarrow \mathbb{R}^d$  of the one-step integrator (3) corresponding to the IRK scheme, defined by (4)–(5), is rewritten in [1] in the following equivalent form:

$$\Phi(y, t, h) := y + \sum_{i=1}^s L_i,$$

where  $L_i \in \mathbb{R}^d$ ,  $i = 1, \dots, s$  are implicitly defined as functions of  $(t, y, h) \in \mathbb{R}^{d+2}$  by

$$L_i = h b_i f(t + c_i h, y + \sum_{j=1}^s \mu_{ij} L_j), \quad i = 1, \dots, s, \quad (30)$$

where

$$\mu_{ij} = a_{ij}/b_j, \quad 1 \leq i, j \leq s.$$

The symplecticity condition (14) is equivalent to

$$\mu_{ij} + \mu_{ji} - 1 = 0, \quad 1 \leq i, j \leq s. \quad (31)$$

The main advantage of the proposed formulation over the standard one is that the absence of multiplications in the symplecticity condition (31) makes possible to find machine number approximations  $\mu_{ij}$  of  $a_{ij}/b_j$  satisfying exactly the symplecticity condition (31).

With that alternative formulation, the Newton iteration reads as follows: Initialize  $L_i^{[0]} = 0$  for  $i = 1, \dots, s$ , and compute for  $k = 1, 2, \dots$

$$1) \quad Y_i^{[k]} := y + \sum_{j=1}^s \mu_{ij} L_j^{[k-1]}, \quad i = 1, \dots, s. \quad (32)$$

$$g_i^{[k]} := -L_i^{[k-1]} + h b_i f(t + c_i h, Y_i^{[k]}), \quad i = 1, \dots, s,$$

2) Solve  $\Delta L_i^{[k]}$  from

$$\Delta L_i^{[k]} - h b_i J_i^{[k]} \sum_{j=1}^s \mu_{ij} \Delta L_j^{[k]} = g_i^{[k]}, \quad i = 1, \dots, s, \quad (33)$$

$$\text{where } J_i^{[k]} = \frac{\partial f}{\partial y}(t + c_i h, Y_i^{[k]}) \quad \text{for } i = 1, \dots, s,$$

$$3) \quad L^{[k]} := L^{[k-1]} + \Delta L^{[k]}, \quad (34)$$

$$(35)$$

In the simplified version of the Newton iteration where the Jacobian matrices  $J_i^{[k]}$  are replaced by a common approximation  $J$  (say,  $J = \frac{\partial f}{\partial y}(t + h/2, y)$ ), the linear system in (32) is replaced by

$$\Delta L^{[k]} = (I_s \otimes I_d - h BAB^{-1} \otimes J)^{-1} \begin{pmatrix} g_1^{[k]} \\ \vdots \\ g_s^{[k]} \end{pmatrix}, \quad (36)$$

In that case, we need to repeatedly solve systems of the form

$$(I_s \otimes I_d - h BAB^{-1} \otimes J) \Delta L = g, \quad (37)$$

for prescribed  $g \in \mathbb{R}^{sd}$ . Repeated solution of linear systems of this form is also required if the linear system in (32) is iteratively solved as described in Subsection 4.2 below.

Of course, (37) can be solved by adapting the technique described in Subsections 3.3 and 3.4 for the solutions of systems of the form (12). We next describe, for the symmetric case (i.e., when the symmetry condition (15) holds), the corresponding procedure (with the notation adopted in Subsection 3.4) to compute the solution  $\Delta L$  of (37):

1. LU decompositions:

- Compute the LU decompositions of the  $\mathbb{R}^{d \times d}$  matrices

$$I_d + h^2 \sigma_i^2 J^2, \quad i = 1, \dots, [s/2], \quad (38)$$

- Compute the matrix  $M \in \mathbb{R}^{d \times d}$  given in (29) (recall that  $\sigma_m = 0$  when  $s$  is odd), and obtain its LU decomposition.

2. Solution of system (37):

- Compute  $R \in \mathbb{R}^{md}$  from

$$R = (Q_1^T \otimes I_d) g + h (DQ_2^T \otimes J) g,$$

- Compute

$$d = h J \sum_{i=1}^m \alpha_i (I_d + h^2 \sigma_i^2 J^2)^{-1} R_i,$$

- Compute  $\Delta z \in \mathbb{R}^d$  as the solution of the linear system  $M \Delta z = d$ ,
- Next, compute  $W_1, \dots, W_m \in \mathbb{R}^d$  from

$$(I_d + h^2 \sigma_i^2 J^2) W_i - \frac{\alpha_i}{2} J \Delta z = R_i, \quad i = 1, \dots, m.$$

- Follow by computing  $W_{m+1}, \dots, W_s \in \mathbb{R}^d$  from

$$\begin{pmatrix} W_{m+1} \\ \vdots \\ W_s \end{pmatrix} = - \begin{pmatrix} h \sigma_1 J W_1 \\ \vdots \\ h \sigma_{s-m} J W_{s-m} \end{pmatrix} + (Q_2^T \otimes I_d) g.$$

- And finally,  $\Delta L \in \mathbb{R}^{sd}$  is obtained from

$$\Delta L = (BQ \otimes I_d) \begin{pmatrix} W_1 \\ \vdots \\ W_s \end{pmatrix}.$$

## 4 Implementation of symplectic IRK schemes with Newton-like iterations

In this section, we present an algorithm that implements symplectic IRK schemes by making use of the techniques in previous section. Special care is taken to try to reduce the effect of round-off errors by adapting some techniques used in [1] for the implementation of symplectic IRK schemes with fixed point iterations. Our algorithm is intended to be applied with the 64-bit IEEE double precision floating point arithmetic.

### 4.1 Auxiliary techniques

In this subsection we summarize some techniques associated to the use of finite precision arithmetic that we applied in the fixed point iteration implementation of symplectic IRK schemes proposed in [1], and will be used in the algorithm proposed in Subsections 4.3.

Let  $\mathbb{F} \subset \mathbb{R}$  be the set of machine numbers of the 64-bit IEEE double precision floating point arithmetic. We consider the map  $\text{fl} : \mathbb{R} \rightarrow \mathbb{F}$  that sends each real number  $x$  to a nearest machine number  $\text{fl}(x) \in \mathbb{F}$ .

#### 4.1.1 Kahan's compensated summation

The application of any one-step integrator of the form (3) requires computing sums of the form

$$y_{n+1} = y_n + x_n, \quad n = 0, 1, 2, \dots, \quad (39)$$

For an actual implementation that only uses a floating point arithmetic with machine numbers in  $\mathbb{F}$ , special care must be taken with the additions (39). The naive recursive algorithm  $\hat{y}_{n+1} := \text{fl}(\hat{y}_n + \text{fl}(x_n))$ , ( $n = 0, 1, 2, 3 \dots$ ), typically suffers, for large  $n$ , a significant loss of precision due to round-off errors. It is well known that such a round-off error accumulation can be greatly reduced with the use of Kahan's compensated summation algorithm [11] (see also [9][13]).

Given  $y_0 \in \mathbb{R}^d$  and a sequence  $\{x_0, x_1, \dots, x_n, \dots\} \subset \mathbb{F}^d$  of machine numbers, Kahan's algorithm is aimed to compute the sums  $y_n = y_0 + \sum_{\ell=0}^{n-1} x_\ell$ , ( $n \geq 1$ ), using a prescribed floating point arithmetic, more precisely than with the naive

recursive algorithm. The actual algorithm reads as follows:

```

 $\tilde{y}_0 = \text{fl}(y_0); e_0 = \text{fl}(y_0 - \tilde{y}_0);$ 
for  $l \leftarrow 0$  to  $n$  do
     $X_l = \text{fl}(x_l + e_l);$ 
     $\tilde{y}_{l+1} = \text{fl}(\tilde{y}_l + X_l);$ 
     $\hat{X}_l = \text{fl}(\tilde{y}_{l+1} - \tilde{y}_l);$ 
     $e_{l+1} = \text{fl}(X_l - \hat{X}_l);$ 
end

```

**Algorithm 1:** Kahan's compensated summation

The sums  $\tilde{y}_l + e_l \in \mathbb{R}^d$  are more precise approximations of the exact sums  $y_l$  than  $\tilde{y}_l \in \mathbb{F}$ . Algorithm 1 can be interpreted as a family of maps parametrized by  $n$  and  $d$ ,

$$S_{n,d} : \mathbb{F}^{(n+3)d} \rightarrow \mathbb{F}^{2d},$$

that given the arguments  $\tilde{y}_0, e_0, x_0, x_1, \dots, x_n \in \mathbb{F}^d$ , returns  $\tilde{y}_{n+1}, e_{n+1} \in \mathbb{F}^d$  such that  $\tilde{y}_{n+1} + e_{n+1} \approx (\tilde{y}_0 + e_0) + x_0 + x_1 + \dots + x_n$  with some small error.

#### 4.1.2 Stopping criterion for iterative processes

Given a smooth map  $F : \mathbb{R}^D \rightarrow \mathbb{R}^D$  and  $Z^{[0]} = (Z_1^{[0]}, \dots, Z_D^{[0]}) \in \mathbb{R}^D$  assume that the iteration

$$Z^{[k]} = F(Z^{[k-1]}), \quad \text{for } k = 1, 2, \dots \quad (40)$$

produces a sequence  $\{Z^{[0]}, Z^{[1]}, Z^{[2]}, \dots\} \subset \mathbb{R}^D$  that converges to a fixed point  $Z^{[\infty]}$  of  $F$ .

Assume now that instead of the original map  $F$ , we have a computational substitute

$$\tilde{F} : \mathbb{F}^D \rightarrow \mathbb{F}^D. \quad (41)$$

Ideally, for each  $Z \in \mathbb{F}^D$ ,  $\tilde{F}(Z) := \text{fl}(F(Z))$ . In practice, the intermediate computations to evaluate  $\tilde{F}$  are typically made using the floating point arithmetic corresponding to  $\mathbb{F}$ , which will result in some additional error caused by the accumulated effect of several round-off errors.

The resulting sequence  $\tilde{Z}^{[k]} = \tilde{F}(\tilde{Z}^{[k-1]})$ ,  $k = 1, 2, \dots$  (started with  $\tilde{Z}^{[0]} = \text{fl}(Z^{[0]})$ ) will either converge to a fixed point of  $\tilde{F}$  in a finite number  $K$  of iterations or will fail to converge. In the former case, the fixed point  $\tilde{Z}^{[K]} \in \mathbb{F}^D$  of  $\tilde{F}$  may be expected to be a good approximation of the fixed point  $Z^{[\infty]} \in \mathbb{R}^D$  of  $F$ . In the later case, one would expect that there exists an index  $K$  such that the approximations  $\tilde{Z}^{[k]} \approx Z^{[\infty]}$  improves for increasing  $k$  up to  $k = K$ , but the quality of the approximations  $\tilde{Z}^{[k]}$  does not improve for  $k > K$ . It then make

sense to apply an algorithm of the form

```

k = 0;
Z̃[0] = fl(Z[0]);
while ( ContFcn(Z̃[0], …, Z̃[k]) ) do
    | k = k + 1;
    | Z̃[k] = F̃(Z̃[k-1]);
end

```

**Algorithm 2:** Stopping criterion

where  $\text{ContFcn}(\tilde{Z}^{[0]}, \dots, \tilde{Z}^{[k]})$  gives either true if it is judged appropriate to continue iterating, and false otherwise. In [1], we propose defining this function so that  $\text{ContFcn}(\tilde{Z}^{[0]}, \dots, \tilde{Z}^{[k]})$  returns,

$$\left\{ \begin{array}{ll} \text{false} & \rightarrow \text{ if } (\tilde{Z}^{[k]} = \tilde{Z}^{[k-1]}) \text{ or} \\ & \quad k > 1 \text{ and } k = K - 1 \text{ and } k = K \text{ and } \forall j \in \{1, \dots, D\}, \\ & \quad \min(\{|\tilde{Z}_j^{[1]} - \tilde{Z}_j^{[0]}|, \dots, |\tilde{Z}_j^{[k-1]} - \tilde{Z}_j^{[k-2]}|\} / \{0\}) \leq |\tilde{Z}_j^{[k]} - \tilde{Z}_j^{[k-1]}| \\ \text{true} & \rightarrow \text{ otherwise.} \end{array} \right. \quad (42)$$

The output  $\tilde{Z}^{[K]}$  of the algorithm will be a fixed point of  $\tilde{F}$  when it stops because  $\tilde{Z}^{[K]} = \tilde{Z}^{[K-1]}$ , and in any case it is not expected that  $\tilde{Z}^{[k]}$  for  $k > K$  be a better approximation of the fixed point  $Z^{[\infty]} \in \mathbb{R}^D$  of  $F$  than  $\tilde{Z}^{[K]}$ .

## 4.2 An inexact Newton iteration

In our implementation of symplectic IRK schemes to be described in Subsection 4.3, we consider a modified version of the Newton iteration (32). In each iteration, instead of exactly solving  $\Delta L_i^{[k]}$  from linear system of the form

$$\Delta L_i^{[k]} - hb_i J_i \sum_{j=1}^s \mu_{ij} \Delta L_j^{[k]} = g_i^{[k]}, \quad i = 1, \dots, s, \quad (43)$$

where

$$g_i^{[k]} = -L_i^{[k-1]} + hb_i f\left(t + c_i h, y + \sum_{j=1}^s \mu_{ij} L_j^{[k-1]}\right), \quad i = 1, \dots, s, \quad (44)$$

and

$$\Delta L^{[k]} = \begin{pmatrix} \Delta L_1^{[k]} \\ \vdots \\ \Delta L_s^{[k]} \end{pmatrix} \in \mathbb{R}^{sd}, \quad g^{[k]} = \begin{pmatrix} g_1^{[k]} \\ \vdots \\ g_s^{[k]} \end{pmatrix} \in \mathbb{R}^{sd},$$

we iteratively compute a sequence  $\Delta L_i^{[k,0]}, \Delta L_i^{[k,1]}, \Delta L_i^{[k,2]}, \dots$  of approximation of its solution  $\Delta L^{[k]} \in \mathbb{R}^{sd}$  as shown below (Algorithm 3).

```

 $\Delta L^{[k,0]} = (I_s \otimes I_d - h BAB^{-1} \otimes J)^{-1} g^{[k]};$ 
while ( ContFcn(fl32( $\Delta L^{[k,0]}$ ),  $\dots$ , fl32( $\Delta L^{[k,\ell]}$ )) ) do
     $l = l + 1;$ 
     $G_i^{[k,\ell]} = g_i^{[k]} - \Delta L_i^{[k,\ell-1]} + hb_i J_i \sum_{j=1}^s \mu_{ij} \Delta L_j^{[k,\ell-1]}, \quad i = 1, \dots, s;$ 
     $\Delta L^{[k,\ell]} = \Delta L^{[k,\ell-1]} + (I_s \otimes I_d - h BAB^{-1} \otimes J)^{-1} G^{[k,\ell]};$ 
end

```

**Algorithm 3:** Inner iteration

where  $\text{fl}_{32}(x)$  represents the 32-bit IEEE single precision machine number that is closest to  $x \in \mathbb{R}$ , and we let  $\text{fl}_{32}$  act componentwise on vectors.

In the algorithm we propose in Subsection 4.3, the Jacobian matrices  $J_i$  in (43) will be evaluated in approximations of the stage values  $Y_i$  that are accurate at the single precision level. This implies that it does not make sense to apply the iteration (Algorithm 3) until an accurate double precision approximation  $\Delta L^{[k,\ell]}$  of the solution  $\Delta L^{[k]}$  of (43) is obtained. Motivated by that, we will stop the iteration when  $\text{ContFcn}(\text{fl}_{32}(\Delta L^{[k,0]}), \dots, \text{fl}_{32}(\Delta L^{[k,\ell]}))$  returns false (i.e., typically, when  $\text{fl}_{32}(\Delta L^{[k,\ell]}) = \text{fl}_{32}(\Delta L^{[k,\ell-1]})$ ).

### 4.3 Algorithm for one step of the IRK scheme

In our implementation, the numerical solution  $y_n \approx y(t+hn) \in \mathbb{R}^d$ ,  $n = 1, 2, \dots$ , is obtained as the sum  $\tilde{y}_n + e_n$  of two vectors in  $\mathbb{F}^d$ . In particular, the initial value  $y_0 \in \mathbb{R}^d$  is (approximately) represented as  $\tilde{y}_0 + e_0$ , where  $\tilde{y}_0 = \text{fl}(y_0)$  and  $e_0 = \text{fl}(y_n - \tilde{y}_n)$ . Instead of (3), we actually have

$$(\tilde{y}_{n+1}, e_{n+1}) = \tilde{\Phi}(\tilde{y}_n, e_n, t_n, t_{n+1} - t_n),$$

where  $\tilde{\Phi} : \mathbb{F}^{2d+2} \rightarrow \mathbb{F}^{2d}$ .

Our proposed implementation of one step

$$(\tilde{y}^*, e^*) = \tilde{\Phi}(\tilde{y}, e, t, h)$$

of the IRK scheme is performed in five substeps:

1. Starting from  $L^{[0]} = 0 \in \mathbb{R}^{sd}$ , we apply several simplified Newton iterations (i.e., the simplified version of Newton iterations (32) where the linear system is replaced by (36)) to compute

$$L^{[1]} = L^{[0]} + \Delta L^{[1]}, \quad L^{[2]} = L^{[1]} + \Delta L^{[2]}, \dots$$

until  $\text{fl}_{32}(L^{[k]}) = \text{fl}_{32}(L^{[k-1]})$  (or rather, by using the notation introduced in paragraph 4.1.2, until  $\text{ContFcn}(\text{fl}_{32}(L^{[0]}), \dots, \text{fl}_{32}(L^{[k]}))$  returns false).

2. Use  $L^{[k]}$  to compute the Jacobian matrices

$$J_i = \frac{\partial f}{\partial y} \left( t + c_i h, \tilde{y} + \sum_{j=1}^s \mu_{ij} L_j^{[k]} \right), \quad i = 1, \dots, s,$$

3. Then consider the increment  $\Delta L^{[k]} \in \mathbb{R}^{sd}$  obtained in first substep as an approximation  $\Delta L^{[k,0]}$  of the exact solution  $\Delta L^{[k]}$  of the linear system in (32), and apply the inner iterations (Algorithm 3) to obtain as output an approximation  $\Delta L^{[k,\ell]}$  (accurate at least at single precision level) of the solution  $\Delta L^{[k]}$  of (12).
4. Follow by updating  $L^{[k]} = L^{[k-1]} + \Delta L^{[k,\ell]}$ , and  $k = k + 1$ , and applying a final inexact Newton iteration with the Jacobian matrices  $J_i$  computed in the second substep. More precisely, compute an approximation  $\Delta L^{[k,\ell]}$  (again accurate at least at single precision level) of the solution  $\Delta L^{[k]}$  of (43)–(44) by applying Algorithm 3.
5. Finally, the increment  $\tilde{\Phi}(\tilde{y}, e, t, h)$ , defined as the sum  $(\tilde{y}+e) + \sum_{i=1}^s (L_i^{[k-1]} + \Delta L^{[k,\ell]})$  is accurately obtained as the (unevaluated) sum of the double precision vectors  $\tilde{y}^*, e^* \in \mathbb{F}^d$  with the help of Kahan’s compensated summation algorithm (summarized in paragraph 4.1.1) as follows: First, perform the sum  $\delta := e + \sum_{i=1}^s \Delta L^{[k,\ell]}$  of the vectors with relatively smaller size in the double precision floating point arithmetic, and then compute  $(\tilde{y}^*, e^*) = S_{s,d}(\tilde{y}, \delta, L_1^{[k-1]}, \dots, L_s^{[k-1]})$ .

Some remarks about our actual implementation are in order:

- All the linear system with coefficient matrix  $(I_s \otimes I_d - hBAB^{-1} \otimes J)$  are solved by means of the algorithm at the end of Section 3.
- The coefficients  $\mu_{ij}$  are machine numbers in  $\mathbb{F}$  (i.e., in the target precision floating point system) satisfying exactly the symplecticity condition (31) and the symmetry conditions  $\mu_{j,i} = \mu_{s+1-i, s+1-j}$ .
- The remainders (44) ( $i = 1, \dots, s, k \geq 1$ ) should in principle be computed with  $y \in \mathbb{R}^d$  replaced by  $\tilde{y} + e$  ( $\tilde{y}, e \in \mathbb{F}^d$ ). However, the effect of ignoring the extra digits of  $y$  that may be contained in  $e$  is expected to be so small that it should be enough to take it into account only in the final inexact Newton iteration (substep 4 above). That is, it should be enough considering (44) with  $y \in \mathbb{R}^d$  replaced by  $\tilde{y} \in \mathbb{F}^d$  in all the Newton-like iterations with the exception of the final one. And in the final inexact Newton iteration, rather than computing (44) with  $y \in \mathbb{R}^d$  replaced by  $\tilde{y} + e$ , we make use of the Jacobian matrices  $J_i$  to obtain the following approximation

$$\begin{aligned}
h b_i f\left(t + c_i h, \tilde{y} + e + \sum_{j=1}^s \mu_{ij} L_j^{[k-1]}\right) - L_i^{[k-1]} &\approx \\
\left(h b_i f_i^{[k]} - L_i^{[k-1]}\right) + h b_i J_i e, &
\end{aligned}$$

where  $f_i^{[k]} = f\left(t + c_i h, \tilde{y} + \sum_{j=1}^s \mu_{ij} L_j^{[k-1]}\right)$ .



- If the FMA (fused-multiply-add) instruction is available, it should be used to compute  $hb_i f_i^{[k]} - L_i^{[k-1]}$  (with precomputed coefficients  $hb_i \in \mathbb{F}$  satisfying the symmetry conditions  $hb_{s+1-i} = hb_i$ ).

Our final implementation is summarized in Algorithm 4.

```

 $L^{[0]} = 0;$ 
 $J = \frac{\partial f}{\partial y}(t + h/2, \tilde{y});$ 
 $M = I_d + J \frac{h}{2} \sum_{i=1}^m \alpha_i^2 (I_d + h^2 \sigma_i^2 J^2)^{-1};$ 
Compute the LU decomposition of M;
/***** 1st substep *****/;
 $k = 0;$ 
while ContFcn(fl32( $L^{[0]}$ ), ..., fl32( $L^{[k]}$ ))) do
     $k = k + 1;$ 
 $Y_i^{[k]} = \tilde{y} + \sum_{j=1}^s \mu_{ij} L_j^{[k-1]}, \quad i = 1, \dots, s;$ 
 $f_i^{[k]} = f(t + c_i h, Y_i^{[k]}), \quad i = 1, \dots, s;$ 
 $g_i^{[k]} = h b_i f_i^{[k]} - L_i^{[k-1]}, \quad i = 1, \dots, s;$ 
 $\Delta L^{[k]} = (I_s \otimes I_d - h BAB^{-1} \otimes J)^{-1} g^{[k]};$ 
 $L^{[k]} = L^{[k-1]} + \Delta L^{[k]};$ 
end
/***** 2nd substep *****/;
 $J_i = \frac{\partial f}{\partial y}(t + c_i h, \tilde{y} + \sum_{j=1}^s \mu_{ij} L_j^{[k]}), \quad i = 1, \dots, s;$ 
/***** 3rd substep *****/;
 $\ell = 0;$ 
 $\Delta L^{[k,0]} = \Delta L^{[k]};$ 
while ContFcn(fl32( $\Delta L^{[k,0]}$ ), ..., fl32( $\Delta L^{[k,\ell]}$ ))) do
     $\ell = \ell + 1;$ 
 $G_i^{[k,\ell]} = g_i^{[k]} - \Delta L_i^{[k,\ell-1]} + h b_i J_i \sum_{j=1}^s \mu_{ij} \Delta L_j^{[k,\ell-1]}, \quad i = 1, \dots, s;$ 
 $\Delta L^{[k,\ell]} = \Delta L^{[k,\ell-1]} + (I_s \otimes I_d - h BAB^{-1} \otimes J)^{-1} G^{[k,\ell]};$ 
end
 $L^{[k]} = L^{[k-1]} + \Delta L^{[k,\ell]};$ 
/***** 4th substep *****/;
 $k = k + 1;$ 
 $Y_i^{[k]} = \tilde{y} + \sum_{j=1}^s \mu_{ij} L_j^{[k-1]}, \quad i = 1, \dots, s;$ 
 $f_i^{[k]} = f(t + c_i h, Y_i^{[k]}), \quad i = 1, \dots, s;$ 
 $g_i^{[k]} = (h b_i f_i^{[k]} - L_i^{[k-1]}) + h b_i J_i e, \quad i = 1, \dots, s;$ 
 $\ell = 0;$ 
 $\Delta L^{[k,0]} = (I_s \otimes I_d - h BAB^{-1} \otimes J)^{-1} g^{[k]};$ 
while ContFcn(fl32( $\Delta L^{[k,0]}$ ), ..., fl32( $\Delta L^{[k,\ell]}$ ))) do
     $\ell = \ell + 1;$ 
 $G_i^{[k,\ell]} = g_i^{[k]} - \Delta L_i^{[k,\ell-1]} + h b_i J_i \sum_{j=1}^s \mu_{ij} \Delta L_j^{[k,\ell-1]}, \quad i = 1, \dots, s;$ 
 $\Delta L^{[k,\ell]} = \Delta L^{[k,\ell-1]} + (I_s \otimes I_d - h BAB^{-1} \otimes J)^{-1} G^{[k,\ell]};$ 
end
/***** 5th substep *****/;
 $\delta = e + \sum_{i=1}^s \Delta L_i^{[k,\ell]};$ 
 $(\tilde{y}^*, e^*) = S_{s,d}(\tilde{y}, \delta, L_1^{[k-1]}, \dots, L_s^{[k-1]});$ 

```

**Algorithm 4:** Implementation of one step of the IRK scheme

## 5 Numerical experiments

We next report some numerical experiments to assess our implementation of the 6-stage Gauss collocation method of order 12 based on Newton-like iterations (Algorithm 4) with 64-bit IEEE double precision floating point arithmetic.

### 5.1 The double pendulum stiff problem

We consider the planar double pendulum stiff problem: a double bob pendulum with masses  $m_1$  and  $m_2$  attached by rigid massless rods of lengths  $l_1$  and  $l_2$  and spring of elastic constant  $k$  between both rods (the rods are aligned at equilibrium). For  $k = 0$ , the problem is non-stiff, and the system's stiffness arises through increasing the value of  $k$ .

The configuration of the pendulum is described by two angles  $q = (\phi, \theta)$  : while  $\phi$  is the angle (with respect to the vertical direction) of the first bob with, the second bob's angle is defined by  $\psi = \phi + \theta$ . We denote the corresponding momenta as  $p = (p_\phi, p_\theta)$ .

Its Hamiltonian function  $H(q, p)$  is

$$\begin{aligned} & - \frac{l_1^2 (m_1 + m_2) p_\theta^2 + l_2^2 m_2 (p_\theta - p_\phi)^2 + 2 l_1 l_2 m_2 p_\theta (p_\theta - p_\phi) \cos(\theta)}{l_1^2 l_2^2 m_2 (-2 m_1 - m_2 + m_2 \cos(2\theta))} \\ & - g \cos(\phi) (l_1 (m_1 + m_2) + l_2 m_2 \cos(\theta)) + g l_2 m_2 \sin(\theta) \sin(\phi) + \frac{k}{2} \theta^2, \end{aligned} \quad (45)$$

We consider the following fixed parameters values

$$g = 9.8 \frac{m}{sec^2}, \quad l_1 = 1.0 m, \quad l_2 = 1.0 m, \quad m_1 = 1.0 kg, \quad m_2 = 1.0 kg,$$

and we choose different values for the elastic constant  $k$  to study different levels of stiffness in the double pendulum. The initial values are chosen as follows: For  $k = 0$ , we choose the initial values considered in [1] that gives rise to a non-chaotic trajectory,  $q(0) = (1.1, -1.1)$  and  $p(0) = (2.7746, 2.7746)$ . The initial values for  $k \neq 0$  are chosen as

$$q(0) = \left( 1.1, \frac{-1.1}{\sqrt{1 + 100k}} \right), \quad p(0) = (2.7746, 2.7746)$$

so that the total energy of the system is bounded as  $k \rightarrow \infty$ .

All the integrations are performed with step-size  $h = 2^{-7}$ , that is small enough for round-off errors to dominate over truncation errors in the non-stiff case  $k = 0$ . The truncation errors dominate over the round-off errors for large enough stiffness constant  $k > 0$ . We have integrated over  $T_{end} = 2^{12}$  seconds and sample the numerical results every  $m = 2^{10}$  steps.

## 5.2 Round-off error propagation

First, we check the good performance of round-off error propagation of our new implementation based on Newton-like iterations. In [1] we proposed an implementation based on fixed-point iterations for non-stiff problems that takes special care of reducing the propagation of round-off errors. We will compare the round-off error of both implementations of the 6-stage Gauss collocation method.

We have studied in detail the errors in energy of the double pendulum problem for three values of  $k$ :  $k = 0$ , where the round-off errors dominate over truncation errors,  $k = 2^{10}$ , where both kinds of errors are similar in size, and  $k = 2^{12}$ , where truncation errors dominate over round-off errors. In order to make a more robust comparison of the numerical errors due to round-off errors, we adopt (as in [7]) an statistical approach. We have considered for each of the three initial value problems,  $P = 1000$  perturbed initial values by randomly perturbing each component of the initial values with a relative error of size  $\mathcal{O}(10^{-6})$ .

The numerical tests in Figure 1 seem to confirm the good performance of round-off error propagation of our new implementation. From one hand, one can observe that, as in [1], the fixed-point implementation exhibits a small linear drift of the mean energy error for  $k = 0$  and  $k = 2^{10}$ , while in the Newton implementation this energy drift does not appear at all. On the other hand, the standard deviation of the energy errors are of similar size and grow proportionally to  $t^{\frac{1}{2}}$  in both implementations.

## 5.3 Fixed-point versus Newton iteration

We summarize in Table 1 the main results of numerical integrations for both implementations: the fixed-point iteration and Newton-like iteration for four different values of  $k$ .

We have compared their efficiency by sequential execution of each iteration method, and reported the cpu-time of each numerical integration. In addition, we have reported the number of iterations per step (It. per step) in both implementations and the number of linear systems solved in the Newton implementation. To check the precision of the numerical solution, we have reported the maximum relative energy error,

$$\max \left| \frac{E(t_i) - E(t_0)}{E(t_0)} \right|, \quad t_i = t_0 + ih, \quad i = 0, \dots, \text{steps}.$$

We can see that for low values of  $k$ , the fixed-point implementation is more efficient than Newton implementation. But as we increase the stiffness of the double pendulum, the number of iteration needed at each step in the fixed-point implementation grows up notably, while in the Newton implementation the number of iterations even becomes slightly lower for higher values of  $k$ . Hence, the Newton implementation eventually becomes more efficient as the

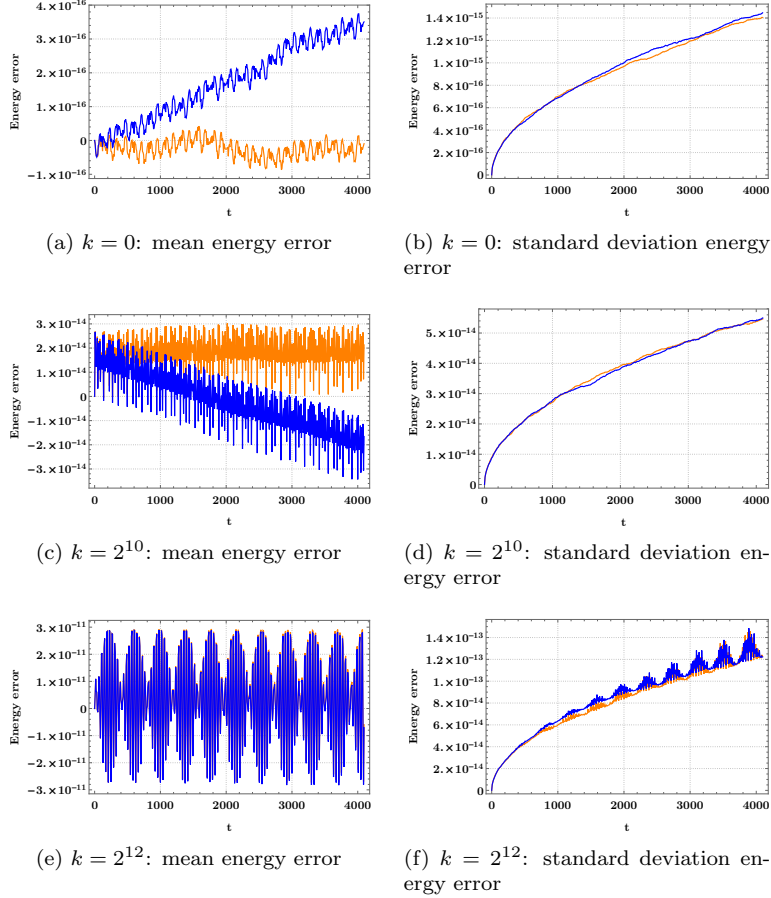


Figure 1: Evolution of mean (left) and standard deviation (right) of relative errors in energy for fixed-point implementation (blue), and Newton implementation (orange). Non-stiff case  $k = 0$  (a,b), one stiff case  $k = 2^{10}$  (c,d) and second stiff case  $k = 2^{12}$  (e,f)

stiffness increases. For  $k$  values higher than  $k = 2^{18}$ , the fixed-point iteration fails to converge, while the Newton implementations succeeds while keeping approximately the same number of iterations per step (cpu-time: 17s; iterations per step: 4.95; Linear solves per step: 10.94).

## 6 Conclusions

Our main contribution is a technique to solve efficiently the simplified linear systems of symplectic IRK schemes. This technique can be adapted for some symmetric non-symplectic schemes as well. Such technique could also be ex-

Table 1: Summary of numerical integrations with fixed-point iteration and Newton iteration based implementation for the following spring's elastic values:  $k = 0$ ,  $k = 2^6$ ,  $k = 2^{12}$  and  $k = 2^{16}$ .  $E_0$  indicates the initial energy of the system. We show the cpu-time, the number of iteration per step (It. per step), the number of linear system solving operations (L. solves per step) and maximum energy error for each numerical computation

k	0	$2^6$	$2^{12}$	$2^{16}$
$E_0$	-14.39	-5.75	-5.64	-5.64
<hr/>				
Fixed-points it.				
Cpu-time (sec.)	10	12	19	51
It. per step	8.58	11.1	22.	64.2
Max energy error	$2.96 \times 10^{-15}$	$1.81 \times 10^{-14}$	$2.94 \times 10^{-11}$	$6.33 \times 10^{-5}$
<hr/>				
Newton it.				
Cpu-time (sec.)	18	20	19	18
It. per step	5.09	5.53	5.58	5.01
L. solves per step	11.37	12.92	12.72	11.04
Max energy error	$1.6 \times 10^{-15}$	$1.74 \times 10^{-14}$	$2.94 \times 10^{-11}$	$6.33 \times 10^{-5}$

exploited for the numerical solution of boundary value problems with collocation methods with Gaussian quadrature nodes.

In addition, an efficient algorithm for implementing symplectic IRK methods with reduced round-off error propagation is provided. A C-code with our implementation for  $s$ -stage Gauss collocation method of order  $2s$  in the 64-bit IEEE double precision floating point arithmetic can be downloaded from IRK-Newton Github software repository or go to the next url: <https://github.com/mikelehu/IRK-Newton>.

**Acknowledgements** M. Antoñana, J. Makazaga, and A. Murua have received funding from the Project of the Spanish Ministry of Economy and Competitiveness with reference MTM2016-76329-R (AEI/FEDER, EU), from the project MTM2013-46553-C3-2-P from Spanish Ministry of Economy and Trade, and as part of the Consolidated Research Group IT649-13 by the Basque Government.

## References

- [1] Antoñana, M., Makazaga, J., Murua, A.: Reducing and monitoring round-off error propagation for symplectic implicit runge-kutta schemes. *Numerical Algorithms* pp. 1–20 (2017). DOI 10.1007/s11075-017-0287-z
- [2] Baboulin, M., Buttari, A., Dongarra, J., Kurzak, J., Langou, J., Langou, J., Luszczek, P., Tomov, S.: Accelerating scientific computations with mixed precision algorithms. *Computer Physics Communications* **180**(12), 2526–2533 (2009). DOI dx.doi.org/10.1016/j.cpc.2008.11.005
- [3] Bickart, T.A.: An efficient solution process for implicit runge-kutta methods. *SIAM Journal on Numerical Analysis* **14**(6), 1022–1027 (1977). DOI 10.1137/0714069
- [4] Brugnano, L., Caccia, G.F., Iavernaro, F.: Efficient implementation of gauss collocation and hamiltonian boundary value methods. *Numerical Algorithms* **65**(3), 633–650 (2014). DOI 10.1007/s11075-014-9825-0
- [5] Butcher, J.C.: On the implementation of implicit runge-kutta methods. *BIT Numerical Mathematics* **16**(3), 237–240 (1976). DOI 10.1007/BF01932265
- [6] Hairer, E., Lubich, C., Wanner, G.: *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, vol. 31. Springer Science & Business Media (2006). DOI 10.1007/3-540-30666-8
- [7] Hairer, E., McLachlan, R.I., Razakarivony, A.: Achieving brouwer’s law with implicit runge-kutta methods. *BIT Numerical Mathematics* **48**(2), 231–243 (2008). DOI 10.1007/s10543-008-0170-3
- [8] Hairer, E., Murua, A., Sanz-Serna, J.M.: The non-existence of symplectic multi-derivative runge-kutta methods. *BIT Numerical Mathematics* **34**(1), 80–87 (1994). DOI 10.1007/BF01935017
- [9] Higham, N.J.: *Accuracy and stability of numerical algorithms*. Siam (2002). DOI 10.1137/1.9780898718027
- [10] Jay, L.O.: Preconditioning of implicit runge-kutta methods. *Scalable Computing: Practice and Experience* **10** (2009)
- [11] Kahan, W.: Further remarks on reducing truncation errors. *Communications of the ACM* **8**(1), 40 (1965)
- [12] Liniger, W., Willoughby, R.A.: Efficient integration methods for stiff systems of ordinary differential equations. *SIAM Journal on Numerical Analysis* **7**(1), 47–66 (1970). DOI 10.1137/0707002

- [13] Muller, J., Brisebarre, N., De Dinechin, F., Jeannerod, C., Lefevre, V., Melquiond, G., Revol, N., Stehlé, D., Torres, S.: Handbook of floating-point arithmetic. Springer Science & Business Media (2009). DOI 10.1007/978-0-8176-4705-6
- [14] Saad, Y.: Iterative methods for sparse linear systems. SIAM (2003). DOI [dx.doi.org/10.1137/1.9780898718003](https://dx.doi.org/10.1137/1.9780898718003).bm
- [15] Sanz Serna, J., Calvo, M.: Numerical Hamiltonian problems. Chapman and Hall (1994)
- [16] Xie, D.: A new numerical algorithm for efficiently implementing implicit runge-kutta methods. Department of Mathematical Sciences. University of Wisconsin, Milwaukee, Wisconsin, USA (2009)