CONJUGATE-SYMPLECTICITY OF EULER–MACLAURIN METHODS AND THEIR IMPLEMENTATION ON THE INFINITY COMPUTER

F. IAVERNA RO †, F. MAZZIA ‡, M.S. MUKH AMETZHANOV §, AND Y.A.D. SERGEYEV †

Abstract. Multi-derivative one-step methods based upon Euler–Maclaurin integration formulae are considered for the solution of canonical Hamiltonian dynamical systems. Despite the negative result that symplecticity may not be attained by any multi-derivative Runge–Kutta methods, we show that Euler–MacLaurin formulae are all topologically conjugate to a symplectic formula. This feature entitles them to play a role in the context of geometric integration and, to make their implementation competitive with the existing integrators, we explore the possibility of computing the underlying higher order derivatives with the aid of the Infinity Computer.

Key words. Ordinary differential equations, Hamiltonian systems, multi-derivative methods, numerical infinitesimals, Infinity Computer

AMS subject classifications. 65L06, 65P10, 65D25

1. Introduction. In the present work, we will consider the application of multi-derivative one-step methods for the numerical solution of canonical Hamiltonian problems

\[ y' = J \nabla H(y), \quad y(t_0) = y_0 \in \mathbb{R}^{2m}, \tag{1.1} \]

with

\[ y = \begin{pmatrix} q \\ p \end{pmatrix}, \quad q, p \in \mathbb{R}^m, \quad J = \begin{pmatrix} O & I \\ -I & O \end{pmatrix}, \tag{1.2} \]

where \( q \) and \( p \) are the generalized coordinates and conjugate momenta, \( H : \mathbb{R}^{2m} \to \mathbb{R} \) is the Hamiltonian function and \( I \) stands for the identity matrix of dimension \( m \). It is well-known that the flow \( \varphi_t : y_0 \to y(t) \) associated with the dynamical system (1.1) is symplectic, namely its Jacobian satisfies

\[ \frac{\partial \varphi_t(y)}{\partial y}^\top J \frac{\partial \varphi_t(y)}{\partial y} = J, \text{ for all } y \in \mathbb{R}^{2m}. \tag{1.3} \]

Symplecticity is a characterizing property of canonical Hamiltonian systems and has relevant implications on the geometric properties of the orbits in the phase space. Consequently, the search of symplectic methods for the numerical integration of (1.1) forms a prominent branch of research. We recall that a one-step method \( y_1 = \Phi_h(y_0) \)
( \( h \) is the stepsize of integration) is called symplectic if its Jacobian matrix is symplectic, i.e., \( \Phi_h \) satisfies the analog of (1.3) with \( \Phi_h(y) \) in place of \( \varphi_t(y) \). The main feature of symplectic integrators is the conservation of all quadratic first integrals of a Hamiltonian system. Though they fail to conserve non quadratic Hamiltonian functions, a backward error analysis shows that, when implemented with constant stepsize and under regularity assumptions, they provide a near conservation of the Hamiltonian over exponentially long times [2] (see also [15, page 366]).

The study of symplecticity in combination with multi-derivative R-K methods was initiated by Lasagni [20] who provided a sufficient algebraic condition for a multi-derivative Runge–Kutta method to be symplectic. The brief investigation culminated with the work of Hairer, Murua, and Sanz Serna [16] who showed that, for irreducible multi-derivative R–K methods, Lasagni’s condition is also necessary but it may only be satisfied by standard R–K formulae.

Given this background, it does make sense to wonder whether one-step multi-derivative formulae may exist which are conjugate to a symplectic method. A method \( y_1 = \Phi_h(y_0) \) is conjugate to a symplectic method \( y_1 = \Psi_h(y_0) \) if a global change of coordinates \( \chi_h(y) = y + O(h) \) exists such that \( \Phi_h = \chi_h \circ \Psi_h \circ \chi_h^{-1} \). We observe that the solution \( \{y_n\} \) of a symplectic conjugate method satisfies \( y_n = \Phi^n_h(y_0) = (\chi_h \circ \Psi_h \circ \chi_h^{-1})^n(y_0) = \chi_h \circ \Psi^n_h \circ \chi_h^{-1}(y_0) \). Consequently, symplectic conjugate methods inherit the long-time behavior of symplectic integrators.

This path of investigation is further motivated by the recent studies concerning the implementation of methods involving higher derivatives of the vector field on the Infinity Computer, a new type of a supercomputer allowing one to work numerically with infinite and infinitesimal numbers [1, 23, 38, 39, 40]. The final goal of this new approach is to improve the computational effort associated with the evaluation of the involved derivatives and make them competitive with more standard integrators. In this paper, the Infinity Computer is used for this purpose. It is based on the positional numeral system with the infinite radix \( \infty \) (called grossone and introduced as the number of elements of the set of natural numbers \( N \)) introduced in [27, 28, 30] (see also recent surveys [32, 34]). The first ideas that can be considered as predecessors to the Infinity Computing and based on the principle "the part is less than the whole" were studied by Bernard Bolzano (see [3] and a detailed analysis in [41]). It should be noted that the Infinity Computing theory is not related either to Cantor’s cardinals and ordinals [6]) or to non-standard analysis [26] or to Levi-Civita field [21].

In the Infinity Computing, with the introduction of \( \infty \) in the mathematical language, all other symbols (like \( \infty \), Cantor’s \( \omega \), \( \aleph_0 \), \( \aleph_1 \), etc.) traditionally used to deal with infinities and infinitesimals in different situations are excluded from the language, because \( \infty \) and other numbers constructed with its help not only can replace all of them but can be used with a higher accuracy. The \( \infty \)-based numeral system avoids indeterminate forms and situations similar to \( \infty + 1 = \infty \) and \( \infty - 1 = \infty \) providing results ensuring that if \( a \) is a numeral written in this numeral system then for any \( a \) (i.e., \( a \) can be finite, infinite, or infinitesimal) it follows \( a + 1 > a \) and \( a - 1 < a \).

To construct a number \( C \) in the \( \infty \)-based numeral system, we subdivide \( C \) into groups corresponding to powers of \( \infty \):

\[
(1.4) \quad C = c_{p_m} \infty^{p_m} + \ldots + c_{p_1} \infty^{p_1} + c_{p_0} \infty^{p_0} + c_{p_{-1}} \infty^{p_{-1}} + \ldots + c_{p_{-k}} \infty^{p_{-k}}.
\]

Then, we can write down the number \( C \) as follows:

\[
(1.5) \quad C = c_{p_m} \infty^{p_m} \ldots c_{p_1} \infty^{p_1} c_{p_0} \infty^{p_0} c_{p_{-1}} \infty^{p_{-1}} \ldots c_{p_{-k}} \infty^{p_{-k}},
\]
where all numerals $c_i \neq 0$ belong to a traditional numeral system and are called *grossdigits*, while numerals $p_i$ are sorted in the decreasing order with $p_0 = 0$

$$p_m > p_{m-1} > \ldots > p_1 > p_0 > p_{-1} > \ldots > p_{-(k-1)} > p_{-k},$$

and called *grosspowers*.

The term having $p_0 = 0$ represents the finite part of $C$ since $c_0 \times 0 = c_0$. Terms having finite positive grosspowers represent the simplest infinite parts of $C$. Analogously, terms having negative finite grosspowers represent the simplest infinitesimal parts of $C$. For instance, the simplest infinitesimal used in this work as the integration step in the Euler method for computing the derivatives is $\frac{1}{\sqrt{\Omega}}$.

The $\Omega$-based methodology has been successfully applied in several areas of Mathematics and Computer Science: single and multiple criteria optimization (see [8, 11, 12]), handling ill-conditioning (see [13, 37]), numerical differentiation and solution of ordinary differential equations (see [1, 38, 40, 31]), cellular automata (see [9, 10]), Euclidean and hyperbolic geometry (see [22]), percolation (see [19, 42]), fractals (see [5, 33, 42]), infinite series and the Riemann zeta function (see [32, 34, 43]), the first Hilbert problem and supertasks (see [24, 29, 34]), Turing machines and probability (see [25, 34, 35, 36]), etc.

The paper is organized as follows. In Section 2 we introduce Euler–Maclaurin methods while in Section 3 we show that they are all conjugate to a symplectic method. Section 4 is devoted to the efficient computation of the derivatives on the Infinity Computer. Finally, some numerical illustration are presented in Section 5 while Section 6 contains some concluding remarks.

2. Euler–Maclaurin methods. Euler–Maclaurin methods are higher derivative collocation methods belonging to the class of Hermite-Obrechkoff methods [17, page 277]. When applied to the general initial value problem

$$y'(t) = f(y(t)), \quad y(t_0) = y_0 \in \mathbb{R}^m,$$

they yield a polynomial $\sigma(t_0 + ch)$ approximating the true solution $y(t)$ in the interval $[t_0, t_0+h]$ ($h$ is the stepsize of integration) defined by means of the following collocation conditions at the ends of the interval

$$\begin{cases} 
\sigma(t_0) = y_0, \\
\sigma^{(j)}(t_0) = y^{(j)}(\sigma(t_0)), \quad j = 1, \ldots, s, \\
\sigma^{(j)}(t_0 + h) = y^{(j)}(\sigma(t_0 + h)), \quad j = 1, \ldots, s. 
\end{cases}$$

The approximation at time $t_1 = t_0 + h$ is then yielded by $y_1 = \sigma(t_0 + h) \simeq y(t_0 + h) + O(h^{p+1})$ with $p = 2s$.

Notice that, on the basis of (2.1), the analytical computation of the $j$-th derivative $y^{(j)}$ involves a tensor of order $j$: for example, for a given $z \in \mathbb{R}^m$, $y''(z) = f'(z)f(z)$, where $f'$ is the Jacobian matrix of the function $f$. This considerably raises the computational cost associated with the implementation of the method as long as higher derivatives are considered. We will see that the use of the Infinity Computer circumvents this issue by producing a precise value of $y^{(j)}(z)$ without explicitly evaluating its analytical expression in terms of the derivatives of $f$.

These integrators derive their name from the well-known Euler–Maclaurin integration formula: if $m$ and $n$ are natural numbers and $g(x)$ with $x \in \mathbb{R}$ is a regular
function defined on \([m, n]\),

\[
\int_{m}^{n} g(x) \, dx = \frac{g(m) + g(n)}{2} + \sum_{i=m+1}^{n-1} g(i)
\]

\begin{equation}
(2.3)
\end{equation}

\[- \sum_{k=1}^{s-1} \frac{B_{2k}}{(2k)!} \left( g^{(2k-1)}(n) - g^{(2k-1)}(m) \right) + R,
\]

where \(s \geq 1\) and \(B_{2k}\) is the \(2k\)-th Bernoulli number. The remainder \(R\) is bounded by

\[
|R| \leq \frac{2}{(2\pi)^{2s-2}} \int_{m}^{n} |g^{(2s-1)}(x)| \, dx.
\]

We now consider the integral form of (2.1) in the interval \([t_0, t_0 + h]\), namely

\[
y(t_0 + ch) = y_0 + h \int_{0}^{c} y'(t_0 + \tau h) \, d\tau \left( = h \int_{0}^{c} f(y(t_0 + \tau h)) \, d\tau \right).
\]

Setting \(c = 1\) and evaluating the integral by means of (2.3) with \(m = 0\) and \(n = 1\) yields

\[
y(t_1) = y_0 + h \int_{0}^{1} y'(t_0 + \tau h) \, d\tau
\]

\[
= y_0 + \frac{h}{2} (f(y_1) + f(y_0)) - \sum_{k=1}^{s-1} \frac{h^{2k}B_{2k}}{(2k)!} \left( y^{(2k)}(y_1) - y^{(2k)}(y_0) \right) + R.
\]

Finally, neglecting the remainder term \(R = O(h^{2s+1})\) provides us with the one-step Euler-Maclaurin formulae

\begin{equation}
(2.4)
y_1 = y_0 + \frac{h}{2} (f(y_1) + f(y_0)) - \sum_{k=1}^{s-1} \frac{h^{2k}B_{2k}}{(2k)!} \left( y^{(2k)}(y_1) - y^{(2k)}(y_0) \right).
\end{equation}

When \(s = 1\), (2.4) becomes the trapezoidal method while, for \(s = 2\) and \(s = 3\) we get the fourth and sixth order methods

\begin{equation}
(2.5)
y_1 = y_0 + \frac{h}{2} (f(y_1) + f(y_0)) - \frac{h^2}{12} (y_1'' - y_0''),
\end{equation}

\begin{equation}
(2.6)
y_1 = y_0 + \frac{h}{2} (f(y_1) + f(y_0)) - \frac{h^2}{12} (y_1'' - y_0'') + \frac{h^4}{720} \left( y_1^{(4)} - y_0^{(4)} \right),
\end{equation}

where, to simplify the notation, we have set \((2k)_{0,1} = y^{(2k)}(y_{0,1})\).

3. Conjugate Symplecticity. In terms of the characteristic polynomials

\[
\rho(z) = z - 1, \quad \sigma(z) = \frac{1}{2} (z + 1)
\]

\footnote{In formula (2.3) we have used \(s - 1\) in place of \(s\) to make the argument consistent with formulae (2.2). When \(s = 1\) we get the standard composite trapezoidal quadrature rule. The first even Bernoulli number are \(B_2 = 1/6, B_4 = -1/30, B_6 = 1/42, B_8 = -1/30, \ldots\).}
formula (2.4) reads, for a generic time $t_n$,

$$
\rho(E) y_n = h\sigma(E) f(y_n) - \sum_{k=1}^{s-1} \frac{h^{2k} B_{2k}}{(2k)!} \rho(E) y^{(2k)}(y_n),
$$

where $E$ denotes the shift operator: $E(y_n) = y_{n+1}$.

To show that (3.1) is conjugate to a symplectic method, we will exploit the following result stated in [7] and subsequently used in [14] to derive the conjugate symplecticity property of symmetric multistep methods.

**Lemma 3.1.** [7] Assume that problem (2.1) admits a quadratic first integral $Q(y) = y^\top S y$ (with $S$ a symmetric matrix) and is solved by a $B$-series integrator $\Phi_h(y)$. The following properties are equivalent:

(a) $\Phi_h(y)$ has a modified first integral of the form $\tilde{Q}(y) = Q(y) + O(h)$;

(b) $\Phi_h(y)$ is formally conjugate to a symplectic $B$-series method.

Guided by the above equivalence, hereafter we show that Euler–MacLaurin methods admit a $B$-series expansion and preserve a modified invariant $\tilde{Q}(y) = Q(y) + O(h)$. Though the former property may be directly deduced from [16], where a $B$-series representation of a generic multi-derivative Runge-Kutta method has been obtained, we prefer to derive it by a direct computation on formula (3.1) which, on one hand, will be later exploited to state property (a) of Lemma 3.1 and, on the other hand, will reveal a close relationship of these formulae with the trapezoidal method.

For an analytic function $u(t)$ and a stepsize $h > 0$, we introduce the shift operator $E_h(u(t)) = u(t + h)$ and and recall the relation ($D$ denotes the derivative operator)

$$
E_h = e^{hD} = \sum_{k=0}^{\infty} \frac{h^k}{k!} D^k.
$$

**Theorem 3.2.** The map $y_1 = \Phi_h(y_0)$ associated with the one-step method (3.1) admits a $B$-series expansion and satisfies property (a) of Lemma 3.1.

**Proof.** From the generating function of Bernoulli numbers (see, for example [18])

$$
\frac{z}{e^z - 1} = \sum_{k=0}^{\infty} B_k \frac{z^k}{k!},
$$

we get, considering that $B_1 = -1/2$ and $B_k = 0$ for $k$ odd,

$$
\frac{z \sigma(e^z)}{\rho(e^z)} = \frac{1}{2} \frac{z(e^z + 1)}{e^z - 1} = \frac{z}{e^z - 1} + \frac{z}{2} = 1 + \sum_{k=1}^{\infty} B_{2k} \frac{z^{2k}}{(2k)!}.
$$

In the spirit of backward analysis, we look for an analytical function $u(t)$ formally satisfying the difference equation (3.1) that is, by virtue of (3.2),

$$
\rho(e^{hD}) u(t) = h\sigma(e^{hD}) f(u(t)) - \sum_{k=1}^{s-1} \frac{B_{2k}}{(2k)!} h^{2k} \rho(e^{hD}) D^{2k-1} f(u(t)).
$$

Multiplying both sides of the previous equation by $D \rho(e^{hD})^{-1}$ yields

$$
\dot{u}(t) = hD \rho(e^{hD})^{-1} \sigma(e^{hD}) f(u(t)) - \sum_{k=1}^{s-1} \frac{B_{2k}}{(2k)!} h^{2k} D^{2k} f(u(t)),
$$
and, by taking into account (3.3), we finally arrive at

\[ \dot{u}(t) = \left( 1 + \sum_{k=s}^{\infty} \frac{B_{2k}}{(2k)!} h^{2k} D^{2k} \right) f(u(t)). \]  

Equation (3.4) coupled with the initial condition \( u(t_0) = y_0 \) is nothing but the modified differential equation associated with the Euler–MacLaurin method of order 2s, so that \( u(t_0 + nh) = y_n \). Expanding the solution of (3.4) in Taylor series, we get

\[ \Phi_h(y_0) = y_1 = u(t_0 + h) = y_0 + hf(y_0) + \sum_{k=s}^{\infty} \frac{B_{2k}}{(2k)!} h^{2k+1} D^{2k} f(y_0) \]

\[ + \frac{h^2}{2!} f'(y_0) f(y_0) + \sum_{k=s}^{\infty} \frac{B_{2k}}{(2k)!} h^{2k+2} D^{2k+1} f(y_0) + \ldots. \]

where \( f'(y) \) is the Jacobian of \( f(y) \) and we have set \( D^r f(y_0) = D^r f(u(t)) \big|_{t=t_0} \). Collecting like powers of \( h \) in the above expression yields a formal power series expansion in the stepsize \( h \), that is a \( B \)-series expansion.

To show that \( \Phi_h(y) \) admits a modified first integral \( \tilde{Q}(y) = Q(y) + O(h^{2s}) \), we follow the same flow of computation appearing in [15, Theorem 4.10 on page 591], that states an analogous property for symmetric linear multistep methods. We first notice that

\[ \left( 1 + \sum_{k=s}^{\infty} \frac{B_{2k}}{(2k)!} h^{2k} D^{2k} \right)^{-1} = 1 + \sum_{k=s}^{\infty} \gamma_k z^{2k}, \]

for suitable coefficients \( \gamma_k \). Thus (3.4) is tantamount to

\[ \left( 1 + \sum_{k=s}^{\infty} \gamma_k h^{2k} D^{2k} \right) \dot{u}(t) = f(u(t)). \]  

Multiplying both sides of (3.5) by the term \( u(t)^\top S \) yields

\[ \frac{1}{2} \frac{d}{dt} Q(u(t)) + \sum_{k=s}^{\infty} \gamma_k h^{2k} u(t)^\top Su^{(2k+1)}(t) = 0, \]

where we have taken into account that \( 2u(t)^\top Su(t) = \tilde{Q}(u(t)) \) and \( z^\top Sf(z) = 0 \) for any \( z \in \mathbb{R}^m \), since \( Q(y) \) is a first integral of the original system (2.1). A repeated use of the property (the explicit dependence on the time \( t \) is omitted to simplify the notation)

\[ u^{(i)^\top} Su^{(j)} = \frac{d}{dt} \left( u^{(i)^\top} Su^{(j-1)} \right) - u^{(i+1)^\top} Su^{(j-1)}, \]

with, in particular,

\[ u^{(i)^\top} Su^{(i+1)} = \frac{1}{2} \frac{d}{dt} \left( u^{(i)^\top} Su^{(i)} \right), \]

allows us to cast each term \( u(t)^\top Su^{(2k+1)}(t) \) in (3.6) as

\[ u^\top Su^{(2k+1)} = \frac{d}{dt} \left( u^\top Su^{(2k)} - \dot{u}^\top Su^{(2k-1)} + \cdots + (-1)^k \frac{1}{2} u^{(k)^\top} Su^{(k)} \right). \]
We observe that the sum in brackets on the right hand side may be formally cast as a function of $u(t)$ by replacing all the derivatives with the aid of the modified differential equation (3.4). After this substitution, denoting by

$$Q_k(u(t)) = 2\gamma_k u(t)^T S u^{(2k+1)}(t)$$

and $\tilde{Q}(u) = Q(u) + \sum_{k=s}^{\infty} h^{2k} Q_k(u)$, from (3.6) we finally obtain $\frac{d}{dt} \tilde{Q}(u(t)) = 0$ which concludes the proof.

4. Computation of the derivatives. One drawback with these implicit methods is the computation of high order derivatives. Symbolic or automatic differentiation are often preferred to finite differences techniques involving terms in $y$ and $y'$. Infact, these latter suffer from numerical instability when the increment becomes small, and requires additional stages. This drawback is overcome on the Infinity Computer and hereafter we illustrate two possible approaches in order to compute the $k$-th derivative of $y(t)$ at time $t_i$.

**Strategy (a).** This strategy was first proposed in [38]. We perform $k$ infinitesimals steps starting at time $t_i$ using the explicit Euler formula with stepsize $\bar{h} = \Theta^{-1}$ as follows:

$$y_{i,1} = y_i + \Theta^{-1} f(y_i), \quad y_{i,2} = y_{i,1} + \Theta^{-1} f(y_{i,1}), \ldots y_{i,k} = y_{i,k-1} + \Theta^{-1} f(y_{i,k-1}).$$

Then, the values of the needed derivatives can be obtained by means of the forward differences $F_k^{[y_i,0, y_i,1, \ldots, y_i,k]}$, with $\bar{h} = \Theta^{-1}$ as follows

$$(4.1) \quad y^{(k)}(t_i) = \frac{F_k^{[y_i,0, y_i,1, \ldots, y_i,k]}}{\Theta^{-k}} + O(\Theta^{-1})$$

where

$$(4.2) \quad F_k^{[y_i,0, y_i,1, \ldots, y_i,k]} = \sum_{j=0}^{k} (-1)^j \binom{k}{j} y_{i,k-j}, \quad y_{i,0} = y_i.$$

As was proven in [38], since the error of the approximation is $O(\Theta^{-1})$, the finite part of the value $F_k^{[y_i,0, y_i,1, \ldots, y_i,k]}$, gives the exact derivative $y^{(k)}(t_i)$. For a more detailed description of the numerical computation of exact derivatives on the Infinity Computer, see [38].

**Strategy (b).** Let us propose another strategy for computing the exact derivatives, where finite differences may be employed directly on the value of $f$ as follows:

$$(4.3) \quad y^{(k)}(t_i) = f^{(k-1)}(y_i) = \frac{F_{k-1}^{[f(y_i,0), f(y_i,1), \ldots, f(y_i,k-1)]}}{\Theta^{-k-1}} + O(\Theta^{-1})$$

Now, let us prove that formulae (4.1) and (4.3) are equivalent.

**Proposition 1.** Let us suppose that for the solution $y(t)$ of the ordinary differential equation (2.1) it is known the value $y_i = y(t_i)$ at the point $t_i$. Then formulae (4.1) and (4.3) are equivalent.

**Proof.** Since formulae (4.1) and (4.3) differ only in the forward differences, then in order to prove the proposition, it is sufficient to demonstrate that the following equation holds

$$(4.4) \quad F_{k-1}^{[y_i,0, y_i,1, \ldots, y_i,k]} = \Theta^{-1} \cdot F_{k-1}^{[f(y_i,0), \ldots, f(y_i,k-1)]},$$
where $y_{i,0} = y_i$. Let us use the mathematical induction to prove it.

The basis of the induction is $k = 2$. By using formulae (4.1)–(4.3) for $y_{i,0}$, $y_{i,1}$, and $y_{i,2}$ we obtain

\[
F_{z-1}^2[y_{i,0}, y_{i,1}, y_{i,2}] = y_{i,2} - 2y_{i,1} + y_{i,0} = y_{i,0} + \Theta^{-1} \cdot (f(y_{i,0}) + f(y_{i,1})) - 2(y_{i,0} + \Theta^{-1}) \cdot f(y_{i,0}) + y_{i,0} = \Theta^{-1} \cdot (f(y_{i,1}) - f(y_{i,0})) = \Theta^{-1} \cdot F_{z-1}^1[f(y_{i,0}), f(y_{i,1})].
\]

Suppose now that (4.4) holds for $k - 1$, $k \geq 3$. We get

\[
F_{z-1}^k[y_{i,0}, ..., y_{i,k}] = F_{z-1}^{k-1}[y_{i,0}, ..., y_{i,k}] - F_{z-1}^{k-1}[y_{i,0}, ..., y_{i,k-1}]
= \Theta^{-1} \cdot F_{z-1}^{k-2}[f(y_{i,1}), ..., f(y_{i,k-1})] - \Theta^{-1} \cdot F_{z-1}^{k-2}[f(y_{i,0}), ..., f(y_{i,k-2})]
= \Theta^{-1} \cdot F_{z-1}^{k-1}[f(y_{i,0}), ..., f(y_{i,k-1})].
\]

This completes the proof. □

The advantage of strategy (b) with respect to strategy (a) is that the use of formula (4.3) in place of (4.1) decreases the computational costs due to the following reasons:

1. It is not necessary to compute the value $y_{i,k}$ for (4.3), whereas it should be calculated in (4.1).
2. All the computations using (4.1) should be performed using the grosspowers up to $-k$. In contrast, formula (4.3) allows us to work with the numbers using only the grosspowers up to $-(k-1)$.

Let us consider the following example from [40] in order to illustrate these issues.

**Example 1.** Let us find the first 3 derivatives $y'(t_0)$, $y''(t_0)$, and $y'''(t_0)$ of the solution $y(t)$ at the point $t_0 = 0$ of the following initial value problem:

\[
(4.5) \quad \frac{dy}{dt} = \frac{y - 2ty^2}{1 + t}, \quad y(t_0) = 0.4,
\]

whose exact solution is

\[
(4.6) \quad y(t) = \frac{1 + t}{2.5 + t^2}.
\]

Differentiating (4.6) we get the exact values of the following derivatives:

\[
y'(t_0) = 0.4, \quad y''(t_0) = -0.32, \quad y'''(t_0) = -0.96.
\]

Now, let us find these derivatives using strategy (a). First, we perform 3 iterations of the Euler method with the integration step $h = \Theta^{-1}$, truncating all values after the grosspower $-3$:

\[
y_1 = y_0 + \Theta^{-1}f(t_0, y_0) = 0.4 + 0.4\Theta^{-1},
\]
\[
y_2 = y_1 + \Theta^{-1}f(t_0 + \Theta^{-1}, y_1) = 0.4 + 0.8\Theta^{-1} - 0.32\Theta^{-2} - 0.32\Theta^{-3},
\]
\[
y_3 = y_2 + \Theta^{-1}f(t_0 + 2\Theta^{-1}, y_2) = 0.4 + 1.2\Theta^{-1} - 0.96\Theta^{-2} - 1.92\Theta^{-3}.
\]
Applying formulae (4.1), (4.2), we obtain

\[ y'(t_0) \approx 3 \cdot F_{x^{-1}}^1[y_0, y_1] = 3 \cdot (y_1 - y_0) \]
\[ = 3 \cdot (0.4 + 0.43^{-1} - 0.4) = 0.4, \]
\[ y''(t_0) \approx 3 \cdot 2 \cdot F_{x^{-1}}^2[y_0, y_1, y_2] = 3 \cdot (y_2 - 2y_1 + y_0) \]
\[ = 3 \cdot (0.4 + 0.323^{-2} - 0.323^{-3} - 2(0.4 + 0.43^{-1}) + 0.4) \]
\[ = -0.32 - 0.323^{-1} = -0.32 + O(3^{-1}), \]
\[ y'''(t_0) \approx 3 \cdot F_{x^{-1}}^3[y_0, y_1, y_2, y_3] = 3 \cdot (y_3 - 3y_2 + 3y_1 - y_0) \]
\[ = 3 \cdot (0.4 + 1.23^{-1} - 0.963^{-2} - 1.923^{-3} - 3(0.4 + 0.43^{-1} - 0.4) \]
\[ = -0.96, \]

from where we can extract the exact values of \( y'(t_0) \), \( y''(t_0) \), and \( y'''(t_0) \) as finite parts

\( 0.4, -0.32 - 0.323^{-1}, \) and \(-0.96\), respectively.

Let us now apply strategy (b). Here, we need to perform \( k - 1 \) iterations of the Euler method, obtaining the values \( y_1 \) and \( y_2 \), truncating them after the grosspower 0:

\[ f(t_0, y_0) = 0.4, \]
\[ f(t_0 + 3^{-1}, y_1) = 0.4 - 0.323^{-1} - 0.323^{-2}, \]
\[ f(t_0 + 23^{-1}, y_2) = 0.4 - 0.643^{-1} - 1.63^{-2}. \]

Applying formulae (4.3), we obtain

\[ y'(t_0) = 3 \cdot F_{x^{-1}}^0[f(t_0, y_0)] = f(t_0, y_0) = 0.4, \]
\[ y''(t_0) \approx 3 \cdot F_{x^{-1}}^1[y_0, y_1] = f(t_0 + 3^{-1}, y_1) \]
\[ = 3 \cdot (0.4 - 0.323^{-1} - 0.323^{-2} - 0.4) = -0.32 - 0.323^{-1} \]
\[ = -0.32 + O(3^{-1}), \]
\[ y'''(t_0) \approx 3 \cdot 2 \cdot F_{x^{-1}}^2[f(t_0, y_0), f(t_0 + 3^{-1}, y_1)] \]
\[ = 3 \cdot (f(t_0 + 3^{-1}, y_2) - 2f(t_0 + 3^{-1}, y_1) + f(t_0, y_0)) \]
\[ = 3 \cdot (0.4 - 0.643^{-1} - 1.63^{-2} - 2(0.4 - 0.323^{-1} - 0.323^{-2}) + 0.4) \]
\[ = -0.96, \]

from where again we can extract the exact values of \( y'(t_0) \), \( y''(t_0) \), and \( y'''(t_0) \) as finite parts

\( 0.4, -0.32 - 0.323^{-1}, \) and \(-0.96\), respectively.

It should be noticed that the value \( y_2 \) cannot be truncated after the grosspower 0 using the first strategy, because the coefficient of \( 3^{-1} \) at the value \( y_2 \) is used also for computing \( y'''(t_0) \). On the contrary, strategy (b) allows us to use the grosspowers up to \(-2\), which decreases the computational cost of the procedures computing the 2--nd and the 3--rd derivatives.

5. **Numerical illustrations.** In the present section, the Euler–MacLaurin formulae of order four and six (2.5) and (2.6) are applied to a few well-known test
problems to highlight their conservation properties. In particular, the long-time behavior of their numerical solutions is compared with that of the numerical solutions computed by the (symplectic) Gauss methods of order four and six. Though Gauss methods generally exhibit a better accuracy for a given stepsize and order, we stress that a fair comparison of the actual performance of the two classes of methods cannot take aside the computational complexity associated with their implementation and, in particular, the effort in solving the underlying nonlinear systems at each step of the integration procedure. In this respect, we notice that while the dimension of the nonlinear systems associated with the Gauss methods is proportional to the number of stages and hence to the considered order, this is not the case for the Euler–Maclaurin formulae, for which the dimension remains the same, i.e. that of the underlying continuous problem, independently of the order. This study is a delicate issue and will be the subject of a future research.

The numerical experiments have been performed using Matlab R2017b. To reduce the computational complexity needed to advance the solution at each integration step, the nonlinear equations (2.5) and (2.6) have been solved by means of a modified Newton method, using the same Jacobian of the trapezoidal scheme, which is appropriate since the neglected terms are $O(h^2)$. In order to preserve the conservation properties, the nonlinear scheme must be iterated to attain the highest possible accuracy in double precision. Moreover, the derivatives have been computed using strategy (b) defined in the previous section. The Infinity Computer Arithmetic has been emulated in a c++ code callable in Matlab through a suitable interface. We stress that this emulator has been only used for the computation of the derivatives, while all the other operations have been performed using the standard double precision floating point arithmetic available in Matlab. We checked the obtained results with those provided by computing the derivatives analytically.

5.1. **Nonlinear pendulum.** As a first example, we consider the dynamics of a pendulum under influence of gravity. It is usually described in terms of the angle $q$ that the pendulum forms with its stable rest position:

$$\ddot{q} + \sin q = 0,$$

where $p = \dot{q}$ is the angular velocity. The Hamiltonian function associated with (5.1) is

$$H(q, p) = \frac{1}{2}p^2 - \cos q.$$
An initial condition \((q_0, p_0)\) such that \(|H(q_0, p_0)| < 1\) gives rise to a periodic solution \(y(t) = (q(t), p(t))^T\) corresponding to oscillations of the pendulum around the straight-down stationary position. In particular, starting at \(y_0 = (q_0, 0)^T\), the period of oscillation may be expressed in terms of the complete elliptical integral of the first kind as

\[
T(q_0) = \int_0^1 \frac{dz}{\sqrt{(1 - z^2)(1 - \sin^2(q_0/2)z^2)}}.
\]

We chose \(q_0 = \pi/2\) to which there corresponds a period \(T = 7.416298709205487\). We used the fourth and sixth order Euler–Maclaurin and Gauss methods with stepsize \(h = T/11\) to integrate the problem over \(4 \times 10^5\) periods. We then computed the errors \(\|y_n - y_0\|_1\) in the solution and \(|H(y_n) - H(y_0)|\) in the energy function at times multiples of the period \(T\), that is for \(n = 11k\), with \(k = 1, 2, \ldots\). Figures 1 and 2 report the obtained results. On the left plot, we can see that the error in the solution as time increases is essentially the same for the fourth-order formulae and quite similar for the sixth-order formulae. A near conservation of the energy function is observable on the right of each figure. The amplitudes of the bounded oscillations are similar for both methods thus confirming the good long-time behavior properties of Euler–Maclaurin formulae for the problem at hand.

5.2. The Kepler problem. This classical problem describes the motion of two bodies subject to Newton’s law of gravitation. As is well-known, the problem is a completely integrable Hamiltonian dynamical system with two degree of freedom (see, for example, [4]). If the origin of the coordinate system is set on one of the two bodies, the Hamiltonian function

\[
H(q_1, q_2, p_1, p_2) = \frac{1}{2}(p_1^2 + p_2^2) - \frac{1}{\sqrt{q_1^2 + q_2^2}},
\]

describes the motion of the other body, namely an ellipse in the \(q_1-q_2\) plane. Taking as initial conditions

\[
q_1(0) = 1 - e, \quad q_2(0) = 0, \quad p_1(0) = 0, \quad p_2(0) = \sqrt{\frac{1+e}{1-e}},
\]

the trajectory describes an ellipse with eccentricity \(e\) and is periodic with period \(T = 2\pi\). Besides the total energy \(H\), further relevant first integrals are the angular
momentum

\[ M(q_1, q_2, p_1, p_2) = q_1p_2 - q_2p_1. \]

and the Lenz vector \( A = (A_1, A_2, A_3)^\top \), whose components are

\[ A_1(q, p) = p_2M(q, p) - \frac{q_1}{||q||^2}, \quad A_2(q, p) = -p_1M(q, p) - \frac{q_2}{||q||^2}, \quad A_3(q, p) = 0. \]

Of the four first integrals \( H, M, A_1, \) and \( A_2 \) only three are independent so, for example, \( A_1 \) can be neglected. Having set \( e = 0.6 \) and \( h = T/200 \), we integrated the problem over \( 10^5 \) periods and computed the error \( ||y_n - y_0||_1 \) in the solution and in the three first integrals at specific times multiples of the period \( T \), that is for \( n = 200k \), with \( k = 1, 2, \ldots \). Figures 3 and 4 report the obtained results for the fourth and sixth order Euler–Maclaurin (solid lines) and Gauss (dashed lines) methods. On the top-left picture is the absolute error of the numerical solution; the top-right picture shows the error in the Hamiltonian function; the error in the angular momentum is drawn in the bottom-left picture while the bottom-right picture concerns the error in the second component of the Lenz vector. As is expected from a symplectic or a conjugate symplectic integrator, we can see a linear drift in the error \( ||y_n - y_0||_1 \) as the time increases. The same linear growth is experienced in the Lenz invariant. Euler–Maclaurin methods assure a near conservation of the Hamiltonian function and angular momentum. This latter quadratic invariant is precisely conserved (up to machine precision) by Gauss methods due to their symplecticity property.

5.3. Fermi-Pasta-Ulam problem. The Fermi-Pasta-Ulam problem models a physical system composed by \( 2m \) unit point masses disposed along a line and chained together by alternating weak nonlinear springs and stiff linear springs.[4, 15]. The force exerted by the nonlinear springs are assumed proportional to the cube of the
Fig. 4. Results for the sixth-order Euler-Maclaurin method (solid lines) and Gauss method (dashed lines) applied to the Kepler problem.

Fig. 5. Error in the Hamiltonian function (5.3) generated by the Euler Maclaurin methods (solid line) and Gauss methods (dashed line) of order 4 (left picture) and order 6 (right picture).

displacement of the associated masses. Denoting by \( q_1, q_2, \ldots, q_{2m} \) the displacements of the masses from their rest points and assuming the endpoints of the external springs to be fixed, \( q_0 = q_{2m+1} = 0 \), the resulting Hamiltonian problem is defined by the energy function

\[
H(q, p) = \frac{1}{2} \sum_{i=1}^{m} (p_{2i-1}^2 + p_{2i}^2) + \frac{\omega^2}{4} \sum_{i=1}^{m} (q_{2i} - q_{2i-1})^2 + \sum_{i=0}^{m} (q_{2i+1} - q_{2i})^4,
\]

where \( p_i = \dot{q}_i, \ i = 1, \ldots, 2m \) are the conjugate momenta, and \( \omega \) is the stiffness coefficient of the linear strings. Following the discussion in [15, page 22], we introduce the energy \( I_i \) associated with the \( i \)th linear spring

\[
I_i = \frac{1}{4} \left( (p_{2i} - p_{2i-1})^2 + \omega^2(q_{2i} + q_{2i-1}) \right).
\]
The total energy \( I(t) = I_1 + \cdots + I_m \) brought by the linear springs satisfies \( I(t) = I(t_0) + O(\omega^{-1}) \), so that it is almost conserved for large values of the stiffness coefficient \( \omega \). In our experiments we choose \( m = 3 \) and \( \omega = 50 \) and integrated the problem on the interval \([0, 400]\) with stepsize \( h = 0.03 \) and the initial values \( p_0 = (0, \sqrt{2}, 0, 0, 0, 0) \) and \( q_0 = (\frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}, 0, 0, 0, 0) \). The two pictures in Figure 5 display the absolute error in the Hamiltonian function (5.3) evaluated along the numerical solution produced by Euler–Maclaurin (solid line) and Gauss (dashed line) methods. On the left are the results for the fourth-order formulae, while on the right are the results for the sixth order formulae. We can see that, for both methods, the Hamiltonian function is nearly conserved. The pictures in Figure 6 suggest that the very same conclusions apply to the nearly conserved quantity \( I(t) \) above defined: there is an exchange of energy among the linear modes but the total energy does not exhibit any drift.

6. Conclusions. This paper studies the conservation properties of Euler–Maclaurin formulae and their implementation on the Infinity Computer. These are a family of multi-derivative one-step methods containing the classical trapezoidal method. As is the case with this latter method, each member of the family is topologically conjugate to a B-series symplectic formula, which makes it suitable for integrating canonical Hamiltonian systems over long times.

A new approach to compute the exact higher order derivatives using numerical infinities and infinitesimals is proposed. This new technique is simple, does not use the analytical expression of the function \( f(y) \), and avoids hard evaluations with tensors related to the function \( f(y) \). A comparison among this new approach and other known techniques, such as automatic differentiation, is beyond the scope of this paper and will be considered in a future work.
REFERENCES

[1] P. Amodio, F. Iavernaro, F. Mazzia, M. S. Mukhametzhanov, and Y. D. Sergeyev, A generalized Taylor method of order three for the solution of initial value problems in standard and infinity floating-point arithmetic, Mathematics and Computers in Simulation, 141 (2016), pp. 24–39.

[2] G. Benettin and A. Giorgilli, On the hamiltonian interpolation of near to the identity symplectic mappings with application to symplectic integration algorithms, J. Statist. Phys., 74 (1994), pp. 1171–1143.

[3] B. Bolzano, Paradoxien des Unendlichen, C.H. Reclam (German original), 1851.

[4] L. Brugnano and F. Iavernaro, Line Integral Methods for Conservative Problems, Monographs and Research Notes in Mathematics, CRC Press, Boca Raton, FL, 2016.

[5] F. Caldarola, The Sierpinski curve viewed by numerical computations with infinities and infinitesimals, Applied Mathematics and Computation, 318 (2018), pp. 321–328.

[6] G. Cantor, Contributions to the founding of the theory of transfinite numbers, Dover Publications, New York, 1955.

[7] P. Chartier, E. Faou, and A. Murua, An algebraic approach to invariant preserving integrators: the case of quadratic and hamiltonian invariants, Numer. Math., 103(4) (2006), pp. 575–590.

[8] M. Cococcioni, M. Pappalardo, and Y. D. Sergeyev, Lexicographic multi-objective linear programming using grossone methodology: Theory and algorithm, Applied Mathematics and Computation, 318 (2018), pp. 298–311.

[9] L. D’Alotto, Cellular automata using infinite computations, Applied Mathematics and Computation, 218(16) (2012), pp. 8077–8082.

[10] L. D’Alotto, A classification of two-dimensional cellular automata using infinite computations, Indian Journal of Mathematics, 55 (2013), pp. 143–158.

[11] S. De Cosmis and R. D. Leone, The use of grossone in mathematical programming and operations research, Applied Mathematics and Computation, 218(16) (2012), pp. 8029–8038.

[12] R. De Leone, Nonlinear programming and grossone: Quadratic programming and the role of constraint qualifications, Applied Mathematics and Computation, 318 (2018), pp. 290–297.

[13] M. Gaudioso, G. Galliombardo, and M. S. Mukhametzhanov, Numerical infinitesimals in a variable metric method for convex nonsmooth optimization, Applied Mathematics and Computation, 318 (2018), pp. 312–320.

[14] E. Hairer, Conjugate-symplecticity of linear multistep methods, J. Comput. Math., 26(5) (2008), pp. 657–659.

[15] E. Hairer, C. Lubich, and G. Wanner, Geometric Numerical Integration. Structure-Preserving Algorithms for Ordinary Differential Equations, Second ed., Springer, Berlin, 2006.

[16] E. Hairer, A. Murua, and J. Sanz-Serna, The non-existence of symplectic multi-derivative runge-kutta methods, BIT, 34(1) (1994), pp. 80–87.

[17] E. Hairer, S. P. Nørsett, and G. Wanner, Solving ordinary differential equations. I. Nonstiff problems, Second ed., Springer Series in Computational Mathematics 8, Springer-Verlag, Berlin, 1993.

[18] E. Hairer and G. Wanner, Analysis by its history, Undergraduate Texts in Mathematics. Readings in Mathematics, Springer-Verlag, New York, 2008.

[19] D. Iudin, Y. Sergeyev, and M. Hayakawa, Infinity computations in cellular automaton forest-fire model, Communications in Nonlinear Science and Numerical Simulation, 20(3) (2015), pp. 861–870.

[20] F. M. Lasagni, Integration methods for Hamiltonian differential equations, unpublished manuscript, 1990.

[21] T. Levi-Civita, Su sui numeri transfiniti, Rend. Acc. Lincei, Series 5a, 113 (1898), pp. 7–91.

[22] M. Margenstern, Fibonacci words, hyperbolic tilings and grossone, Communications in Nonlinear Science and Numerical Simulation, 21(1–3) (2015), pp. 3–11.

[23] F. Mazzia, Y. D. Sergeyev, F. Iavernaro, P. Amodio, and M. S. Mukhametzhanov, Numerical methods for solving ODEs on the Infinity Computer, in Proc. of the 2nd Intern. Conf. “Numerical Computations: Theory and Algorithms”, vol. 1776, AIP Publishing, New York, 2016, p. 090033.

[24] D. Rizza, Supertasks and numeral systems, in Proc. of the 2nd Intern. Conf. “Numerical Computations: Theory and Algorithms”, vol. 1776, AIP Conference Proceedings, New York, 2016. 090005.

[25] D. Rizza, A study of mathematical determination through Bertrand’s Paradox, Philosophia
F. IAVERNARO, F. MAZZIA, M.S. MUKHAMETZHANOV AND YA.D. SERGEYEV

Mathematica, (2017). In press.

[26] A. Robinson, Non-standard Analysis, Princeton Univ. Press, Princeton, 1996.
[27] Y. Sergeyev, Arithmetic of Infinity, Edizioni Orizzonti Meridionali, CS, 2003, 2nd ed. 2013.
[28] Y. Sergeyev, A new applied approach for executing computations with infinite and infinitesimal quantities, Informatica, 19(4) (2008), pp. 567–596.
[29] Y. Sergeyev, Counting systems and the First Hilbert problem, Nonlinear Analysis Series A: Theory, Methods & Applications, 72(3-4) (2010), pp. 1701–1708.
[30] Y. Sergeyev, Lagrange Lecture: Methodology of numerical computations with infinities and infinitesimals, Rendiconti del Seminario Matematico dell’Università e del Politecnico di Torino, 68(2) (2010), pp. 95–113.
[31] Y. Sergeyev, Higher order numerical differentiation on the Infinity Computer, Optimization Letters, 5(4) (2011), pp. 575–585.
[32] Y. Sergeyev, Un semplice modo per trattare le grandezze infinite ed infinitesime, Matematica nella Società e nella Cultura: Rivista della Unione Matematica Italiana, 8(1) (2015), pp. 111–147.
[33] Y. Sergeyev, The exact (up to infinitesimals) infinite perimeter of the Koch snowflake and its finite area, Communications in Nonlinear Science and Numerical Simulation, 31(1–3) (2016), pp. 21–29.
[34] Y. Sergeyev, Numerical infinities and infinitesimals: Methodology, applications, and repercussions on two Hilbert problems, EMS Surveys in Mathematical Sciences, 4(2) (2017), pp. 219–320.
[35] Y. Sergeyev and A. Garro, Observability of Turing machines: A refinement of the theory of computation, Informatica, 21(3) (2010), pp. 425–454.
[36] Y. Sergeyev and A. Garro, Single-tape and multi-tape Turing machines through the lens of the Grossone methodology, Journal of Supercomputing, 65(2) (2013), pp. 645–663.
[37] Y. Sergeyev, D. Kvasov, and M. Mukhametzhanov, On strong homogeneity of a class of global optimization algorithms working with infinite and infinitesimal scales, Communications in Nonlinear Science and Numerical Simulation, 59 (2018), pp. 319–330.
[38] Y. D. Sergeyev, Solving ordinary differential equations by working with infinitesimals numerically on the infinity computer, Applied Mathematics and Computation, 219(22) (2013), pp. 10668–10681.
[39] Y. D. Sergeyev, Numerical infinitesimals for solving ODEs given as a black-box, in Proc. of the International Conference on Numerical Analysis and Applied Mathematics 2014 (ICNAAM-2014), vol. 1648, AIP Publishing, New York, 2015, p. 150018.
[40] Y. D. Sergeyev, M. S. Mukhametzhanov, F. Mazzia, F. Iavernaro, and P. Amodio, Numerical methods for solving initial value problems on the Infinity Computer, International Journal of Unconventional Computing, 12(1) (2016), pp. 3–23.
[41] K. Trlifajová, Bolzano’s infinite quantities, Foundations of Science, (2018). In press.
[42] M. Vita, S. D. Bartolo, C. Fallico, and M. Veltri, Usage of infinitesimals in the Menger’s sponge model of porosity, Applied Mathematics and Computation, 218(16) (2012), pp. 8187–8196.
[43] A. Zhigljavsky, Computing sums of conditionally convergent and divergent series using the concept of grossone, Applied Mathematics and Computation, 218(16) (2012), pp. 8064–8076.