ALBANIAN JOURNAL
OF MATHEMATICS
Volume 4, Number 4, Pages 105–122
ISSN 1930-1235: (2010)

CREATING VARIATIONAL INTEGRATORS WITH A COMPUTER ALGEBRA SYSTEM

CHRISTIAN HELLSTRÖM

Abstract. A library to create (new) variational integrators to arbitrary order by means of a computer algebra system is presented. The library provides an interface to design as well as analyse variational integrators for dynamical systems, either with non-conservative forces or without.

1. Introduction

Simulations of generic dynamical systems can rarely be carried out analytically, for the class of integrable dynamical systems has zero measure in the space of all dynamical systems, hence the need for numerical integration algorithms. There is an abundance of numerical integration algorithms, available either in the public domain or embedded in proprietary software. Probably the most common algorithms implemented and used are the classical one-step methods, including the classical Runge–Kutta algorithm, multi-step methods, such as the Adams–Bashforth–Moulton algorithm, and adaptive methods, to which the Runge–Kutta–Fehlberg and Bulirsch–Stoer algorithms belong. Nowadays, Taylor's method backed by either symbolic or automatic (numerical) differentiation techniques (see e.g. [11]) is increasingly being used for highly accurate computations, although we shall not dwell on these alternative integration techniques in the sequel.

For simulations over relatively short time spans as compared to the intrinsic time scales standard (non-geometric) integrators are often advantageous, as they can be both accurate and fast. Extended computations require a different, 'geometric' approach, as non-geometric methods tend to generate or dissipate energy artificially due to the fact that they do not respect the fundamental geometry of the phase flow, which means that at some point the errors dominate.

For dynamical systems that can be formulated as Hamiltonian systems there exist so-called geometric numerical integrators. These integrators respect the fundamental (differential) geometric structure, which underlies the dynamical evolution of the system. It has been common to design such geometric numerical integrators based on either previous knowledge of classical numerical integration algorithms, such as the (partitioned) Runge–Kutta methods, or (approximate) solutions to the Hamilton–Jacobi equation for transformations near the identity.

2000 *Mathematics Subject Classification.* 65P30 and 65P10 and 65L05 and 37M15.

Key words and phrases. variational integrators and geometric numerical integrators and numerical integration and computer algebra system.

Funded by the European Commission through the Astrodynamics Network under Marie Curie contract number MRTN-CT-2006-035151.

There is, however, a different approach that bypasses many of the difficulties inherent in the design of higher-order versions of these geometric numerical integrators. It relies on the discretization of the action, from which the numerical algorithms can be derived in a straightforward manner [24]. These variational integrators, as they are known throughout the literature, conserve the Poisson structure. Any continuous symmetries present in the original system translate directly to the discretized version, and thus all (equivariant) momentum maps are preserved infinitesimally. For non-integrable systems either the Poisson structure or the total energy can be conserved exactly in numerical simulations [10], so that variational integrators do not generally preserve the energy. It can be shown, though, that these variational integrators remain close to the original dynamical systems [13, 20], so that in practice the energy error is bounded.

An approach to design variational integrators systematically based on higher-order approximations to the discrete action by means of a computer algebra system is covered here, in particular a freely available package named VarInt for the computer algebra system Maple is presented, with which variational integrators of arbitrary order and based on any derivative-free quadrature rule can be created and analysed.

The fundamental concepts from discrete mechanics are reviewed briefly in section 2, after which the various built-in quadrature rules are discussed in section 3. In section 4, the details of the package VarInt are discussed, and numerous examples are given on how to obtain (new) variational integrators for both generic and specific dynamical systems.

2. Variational Integrators

Consider an autonomous Lagrangian $L\colon \mathbf{T}\mathbb{Q} \to \mathbb{R}$, where $\mathbf{T}\mathbb{Q}$ is the tangent bundle of the configuration space $\mathbb{Q}$, on which the generalized coordinates q form a chart. A chart for the tangent bundle is given by $(q, \dot{q})$, where $\dot{q} = \mathrm{d}q/\mathrm{d}t$ with t the time. Here and henceforth it is assumed that the generalized coordinates are at least $\mathcal{C}^2([a,b],\mathbb{R})$, where $t \in [a,b]$. The corresponding action functional reads

$$S[L] = \int_a^b L(q(t), \dot{q}(t))\, \mathrm{d}t, \tag{1}$$

from which the famous Euler–Lagrange equations are retrieved upon requiring stationarity of the action functional for fixed endpoints, that is $\delta S[L] = 0$ with $\delta q(a) = \delta q(b) = 0$.

Instead of deriving the Euler–Lagrange equations from the action and then discretizing the equations of motion, a different approach is used in the case of variational integrators. Here we discretize the action first by choosing an appropriate quadrature formula, and then we can derive the discrete version of the Euler–Lagrange equations, which are commonly known as the discrete Euler–Lagrange (dEL) equations. The resulting integration algorithms preserve the differential geometric structure of these dynamical systems *automatically* [24]. Moreover, the order of the quadrature formula determines the order of the variational integrator.

2.1. **Quadrature.**

To obtain a one-step numerical integration algorithm, introduce a sequence of times $t_k = hk$ with $k = 0, \ldots, N$, at which the Lagrangian is to be evaluated. Here h is a sufficiently small time step. Furthermore, let $q_k \approx q(t_k)$ and $\dot{q}_k \approx \dot{q}(t_k)$ for $k = 0, \ldots, N$, and consider the action between two consecutive

points in time, say t_k and t_{k+1}. Since for a generic dynamical system with a certain Lagrangian we do not know the functional form of the solutions in advance, we may choose an interpolating function, usually a polynomial, in accordance with the quadrature rule on the interval $[t_k, t_{k+1}]$. For a quadrature rule of arbitrary order, we evaluate the Lagrangian at $(s+1) \geq 2$ distinct nodes, so that each time step is subdivided into s substeps t_k^i for $i = 0, \ldots, s\}$. Define $t_k^0 = t_k$ and $t_k^s = t_{k+1}$, and let $t_k^i - t_k^{i-1} = \gamma_i h > 0$ for $i = 1, \ldots, s$, so that $\sum_{i=1}^{s} \gamma_i = 1$. The action becomes a sum of the multipoint discrete Lagrangian L_d, which depends on the time step h:

$$\begin{aligned} S[L] &= \sum_{k=0}^{N-1} \int_{t_k}^{t_{k+1}} L\left(q(t), \dot{q}(t)\right) \mathrm{d}t \\ &\approx \sum_{k=0}^{N-1} L_\mathrm{d}\left(q_k^0, q_k^1, \ldots, q_k^s\right) \\ &= \sum_{k=0}^{N-1} \sum_{i=1}^{s} L_\mathrm{d}^i\left(q_k^{i-1}, q_k^i\right), \end{aligned} \tag{2}$$

where $L_\mathrm{d}^i \colon \mathbb{Q} \times \mathbb{Q} \to \mathbb{R}$; it relates the multipoint discrete Lagrangian to its basic components defined on each segment of 'length' $\gamma_i h$. Notice that the discrete state space $\mathbb{Q} \times \mathbb{Q}$ contains the same amount of information as the tangent bundle of the configuration manifold, for locally $\mathbf{T}\mathbb{Q} \cong \mathbb{Q} \times \mathbb{Q}$.

Let $L_\mathrm{d}^{[k]}$ be shorthand for $L_\mathrm{d}\left(q_k^0, q_k^1, \ldots, q_k^s\right)$, and let D_i denote the derivative with respect to the argument carrying the substep label i, that is $D_i L_\mathrm{d}^{[k]} = \partial L_\mathrm{d}^{[k]} / \partial q_k^i$. Stationarity of the discrete action, that is $S[L_\mathrm{d}] = 0$, for arbitrary variations δq_k^i yields the discrete Euler–Lagrange (dEL) equations:

$$D_0 L_\mathrm{d}\left(q_{k+1}^0, q_{k+1}^1, \ldots, q_{k+1}^s\right) + D_s L_\mathrm{d}\left(q_k^0, q_k^1, \ldots, q_k^s\right) = 0, \tag{3a}$$

$$D_i L_\mathrm{d}\left(q_k^0, q_k^1, \ldots, q_k^s\right) = 0, \qquad i = 1, \ldots, s-1. \tag{3b}$$

These equations determine the one-step (flow) map $\left(q(t_k), \dot{q}(t_k)\right) \mapsto \left(q(t_{k+1}), \dot{q}(t_{k+1})\right)$ of the variational integrator. These equations can also be written as

$$D_i L_\mathrm{d}^i\left(q_k^{i-1}, q_k^i\right) + D_i L_\mathrm{d}^{i+1}\left(q_k^i, q_k^{i+1}\right) = 0$$

for each of the components $i = 1, \ldots, s$. Please notice that for $i = s$, the last term on the left-hand side is $D_0 L_\mathrm{d}^1\left(q_{k+1}^0, q_{k+1}^1\right) = D_0 L_\mathrm{d}^{[k+1]}$ by virtue of the identity $q_k^{i+s} = q_{k+1}^i$.

It is common to write the one-step map in terms of the canonical coordinates and momenta on the cotangent bundle. To do that, we need to find a discrete analogue of the Legendre transformation, or fibre derivative $\mathbb{F}L \colon \mathbf{T}\mathbb{Q} \to \mathbf{T}^\star\mathbb{Q}$, which reads in generalized coordinates

$$\mathbb{F}L \colon (q, \dot{q}) \mapsto \left(q, \frac{\partial L}{\partial \dot{q}}(q, \dot{q})\right). \tag{4}$$

The discretized form of the Legendre transformation involves the endpoints of each time segment, $\mathbb{F}^\pm L_\mathrm{d}^i \colon \mathbb{Q} \times \mathbb{Q} \to \mathbf{T}^\star\mathbb{Q}$:

$$\begin{aligned} \mathbb{F}^+ L_\mathrm{d}^i &\colon \left(q_k^{i-1}, q_k^i\right) \mapsto \left(q_k^i, p_k^i\right) = \left(q_k^i, D_i L_\mathrm{d}^i\left(q_k^{i-1}, q_k^i\right)\right), \\ \mathbb{F}^- L_\mathrm{d}^i &\colon \left(q_k^{i-1}, q_k^i\right) \mapsto \left(q_k^{i-1}, p_k^{i-1}\right) = \left(q_k^{i-1}, -D_{i-1} L_\mathrm{d}^i\left(q_k^{i-1}, q_k^i\right)\right). \end{aligned}$$

In fact, the discrete Euler–Lagrange equations (3) can be written as

$$\mathbb{F}^+ L_{\mathrm{d}}^i \left(q_k^{i-1}, q_k^i\right) = \mathbb{F}^- L_{\mathrm{d}}^{i+1} \left(q_k^i, q_k^{i+1}\right),$$

which implies that the canonical momenta are unique along any solution. Now, the one-step map, written in canonical coordinates and momenta, is

$$p_k^{i-1} = -D_{i-1} L_{\mathrm{d}}^i \left(q_k^{i-1}, q_k^i\right),$$
$$p_k^i = D_i L_{\mathrm{d}}^i \left(q_k^{i-1}, q_k^i\right),$$

for $i = 1, \ldots, s$, or equivalently,

$$p_k = -D_0 L_{\mathrm{d}} \left(q_k^0, q_k^1, \ldots, q_k^s\right), \tag{7a}$$
$$p_{k+1} = D_s L_{\mathrm{d}} \left(q_k^0, q_k^1, \ldots, q_k^s\right), \tag{7b}$$
$$D_i L_{\mathrm{d}} \left(q_k^0, q_k^1, \ldots, q_k^s\right) = 0, \qquad i = 1, \ldots, s-1, \tag{7c}$$

where the last set of $(s-1)$ equations seems 'unaffected' by the Legendre transformation, and remains as in equation (3b). The reason for that is quite intuitive yet profound: on each time interval an interpolatory function approximates the Lagrangian function, so that the discrete Lagrangian becomes a piecewise smooth function. The momentum p_k^i computed with the discrete Legendre transformation $\mathbb{F}^- L_{\mathrm{d}}^{i+1}$ requires the interpolation function on the time segment $\left[t_k^i, t_k^{i+1}\right]$, or data 'from the right' of t_k^i, whereas the same momentum calculated from the transformation $\mathbb{F}^+ L_{\mathrm{d}}^i$ uses the interpolation function on $\left[t_k^{i-1}, t_k^i\right]$, or values 'from the left' of t_k^i. Obviously, the intermediate momenta ($i = 1, \ldots, s-1$) are identical, because the interpolating function used is the same, so that its derivatives from the left and right coincide. In principle, the momenta at the endpoints of each time interval need not be related at all, for the interpolating function merely has to be equal in value in order to have a piecewise smooth discrete Lagrangian. However, the principle of stationary action relates the approximate (discrete) Lagrangian function to the (approximated) integral curves of the dynamical system, which in turn relates these momenta by means of the discrete Euler–Lagrange equations. Therefore, the momenta are unique along any trajectory, and equations (3b) and (7c) are identical in both representations.

All integrators obtained in this way are structure-preserving, that is to say they preserve the Poisson structure of the flow. The (discrete) Lagrangian flow conserves the (discrete) symplectic form as well as any momentum maps associated with (infinitesimal) invariances of the (discrete) action under symmetry operations, as shown by Marsden and West [24]. Obviously for this statement to hold we have to choose the time step h sufficiently small, which depends on the particulars of the problem under consideration. Please observe that the right-hand sides of the discrete Euler–Lagrange equations (3) and their Hamiltonian equivalents (7) depend on the time step h.

2.2. **Non-Conservative Forces.** The variational formalism arises naturally throughout mathematical physics, and can of course be extended (see e.g. [17, 19, 21, 22, 24] for a few possibilities). The main advantage and actual utility of the variational construction of numerical integration algorithms lies in the fact that non-conservative forces can easily be included in a consistent way. N-body simulations in atomic and molecular physics, astrophysics, and chemistry are excellent candidates for these non-conservative variational integrators, as these simulations often become

unstable under time reversal, so that higher-order geometric numerical integrators based on the symmetric composition of lower-order ones are not viable alternatives.

A force is a fibre-preserving map over the identity $F\colon \mathbf{T}\mathbb{Q} \to \mathbf{T}^\star\mathbb{Q}$, which reads $F\colon (q,\dot{q}) \mapsto (q, F(q,\dot{q}))$ in coordinates. In order to include these non-conservative forces in the variational framework, we merely have to replace Hamilton's principle $\delta S[L] = 0$ by the so-called Lagrange–d'Alembert principle:

$$\sum_{k=0}^{N-1} \left[\delta \int_{t_k}^{t_{k+1}} L(q(t), \dot{q}(t))\, \mathrm{d}t + \int_{t_k}^{t_{k+1}} F(q(t), \dot{q}(t)) \cdot \delta q(t)\, \mathrm{d}t \right] = 0. \tag{8}$$

As before, all integrals are approximated by a quadrature rule from $t \in [t_k, t_{k+1}]$, so that the Lagrange–d'Alembert principle becomes

$$\sum_{k=0}^{N-1} \left[\delta \underbrace{\sum_{i=0}^{s} L\left(q_{\mathrm{int}}\left(t_k^i\right), \dot{q}_{\mathrm{int}}\left(t_k^i\right)\right)}_{L_{\mathrm{d}}\left(q_k^0, q_k^1, \ldots, q_k^s\right)} + \sum_{i=0}^{s} \underbrace{F\left(q_{\mathrm{int}}\left(t_k^i\right), \dot{q}_{\mathrm{int}}\left(t_k^i\right)\right) \cdot \delta q_{\mathrm{int}}\left(t_k^i\right)}_{f_k^i \cdot \delta q_k^i = f_k^i\left(q_k^0, q_k^1, \ldots, q_k^s\right) \cdot \delta q_k^i} \right] = 0, \tag{9}$$

where we have written the discrete Lagrangian in terms of $q_{\mathrm{int}}\left(t_k^i\right) = q_{\mathrm{int}}\left(q_k^0, \ldots, q_k^s; t_k^i\right)$, the interpolatory approximation of $q(t)$ for $t \in [t_k, t_{k+1}]$. It is worth mentioning that

$$\delta q_{\mathrm{int}}\left(t_k^i\right) = \sum_{j=0}^{s} \frac{\partial q_{\mathrm{int}}\left(t_k^i\right)}{\partial q_k^j} \delta q_k^j,$$

and that generally $q_{\mathrm{int}}\left(q_k^0, \ldots, q_k^s; t_k^i\right) \neq q_k^i$. Therefore,

$$f_k^i\left(q_k^0, q_k^1, \ldots, q_k^s\right) = \sum_{j=0}^{s} F\left(q_{\mathrm{int}}\left(t_k^j\right), \dot{q}_{\mathrm{int}}\left(t_k^j\right)\right) \cdot \frac{\partial q_{\mathrm{int}}\left(t_k^j\right)}{\partial q_k^i}.$$

In a manner similar to the derivation of the discrete Euler–Lagrange equations (3), the *forced* discrete Euler–Lagrange equations can be shown to be

$$D_0 L_{\mathrm{d}}^{[k+1]} + f_{k+1}^0 + D_s L_{\mathrm{d}}^{[k]} + f_k^s = 0, \tag{10a}$$

$$D_i L_{\mathrm{d}}^{[k]} + f_k^i = 0, \qquad i = 1, \ldots, s-1. \tag{10b}$$

Again, it is possible to write the forced discrete Euler–Lagrange equations in terms of the canonical coordinates and momenta instead. To that end, define the left and right discrete forces $f_{\mathrm{d}}^{i\,\pm}\colon \mathbb{Q}\times\mathbb{Q} \to \mathbb{R}$, $f_{\mathrm{d}}^{i\,-}\left(q_k^{i-1}, q_k^i\right)$ and $f_{\mathrm{d}}^{i\,+}\left(q_k^{i-1}, q_k^i\right)$, respectively, such that $f_k^0 = f_{\mathrm{d}}^{1\,-}\left(q_k^0, q_k^1\right)$, $f_k^s = f_{\mathrm{d}}^{s\,+}\left(q_k^{s-1}, q_k^s\right)$, and $f_k^i = f_{\mathrm{d}}^{i\,+}\left(q_k^{i-1}, q_k^i\right) + f_{\mathrm{d}}^{i+1\,-}\left(q_k^i, q_k^{i+1}\right)$ for $i = 1, \ldots, s-1$. These, in turn, imply that

$$\int_{t_k}^{t_{k+1}} F(q(t), \dot{q}(t)) \cdot \delta q(t)\, \mathrm{d}t \approx \sum_{i=1}^{s} \left[f_{\mathrm{d}}^{i\,-}\left(q_k^{i-1}, q_k^i\right) \delta q_k^{i-1} + f_{\mathrm{d}}^{i\,+}\left(q_k^{i-1}, q_k^i\right) \delta q_k^i \right].$$

Consequently, one finds the that equations (10) can be written as

$$D_i L_{\mathrm{d}}^i\left(q_k^{i-1}, q_k^i\right) + f_{\mathrm{d}}^{i\,+}\left(q_k^{i-1}, q_k^i\right) + D_i L_{\mathrm{d}}^{i+1}\left(q_k^i, q_k^{i+1}\right) + f_{\mathrm{d}}^{i+1\,-}\left(q_k^i, q_k^{i+1}\right) = 0,$$

for $i = 1, \ldots, s$.

The appropriate discrete Legendre transformations for forced dynamical systems are

$$\mathbb{F}^{f+}L_{\mathrm{d}}^{i}\colon\ \left(q_k^{i-1},q_k^{i}\right)\mapsto\left(q_k^{i},p_k^{i}\right)=\left(q_k^{i},D_iL_{\mathrm{d}}^{i}\left(q_k^{i-1},q_k^{i}\right)+f_{\mathrm{d}}^{i+}\right),$$
$$\mathbb{F}^{f-}L_{\mathrm{d}}^{i}\colon\ \left(q_k^{i-1},q_k^{i}\right)\mapsto\left(q_k^{i-1},p_k^{i-1}\right)=\left(q_k^{i-1},-D_{i-1}L_{\mathrm{d}}^{i}\left(q_k^{i-1},q_k^{i}\right)-f_{\mathrm{d}}^{i-}\right),$$

so that $\mathbb{F}^{f+}L_{\mathrm{d}}^{i}\left(q_k^{i-1},q_k^{i}\right)=\mathbb{F}^{f-}L_{\mathrm{d}}^{i+1}\left(q_k^{i},q_k^{i+1}\right)$ with $i=1,\dots,s$ as before. Now, it is without any effort that we can derive the forced discrete Euler–Lagrange equations on the cotangent bundle:

$$p_k=-D_0L_{\mathrm{d}}\left(q_k^0,q_k^1,\dots,q_k^s\right)-f_k^0, \tag{12a}$$
$$p_{k+1}=D_sL_{\mathrm{d}}\left(q_k^0,q_k^1,\dots,q_k^s\right)+f_k^s, \tag{12b}$$
$$D_iL_{\mathrm{d}}\left(q_k^0,q_k^1,\dots,q_k^s\right)+f_k^i=0,\qquad i=1,\dots,s-1. \tag{12c}$$

The functions f_k^i can be computed with the Maple procedures described below, so that we can easily generate higher-order variational integrators that include non-conservative forces in a 'variational' manner, that is in a way that respects the fundamental differential geometric properties of any dynamical system.

3. Quadrature Rules

In principle any integration formula can be used to approximate the discrete action, and thus generate a variational integrator, although some cautionary remarks are in order. First, autonomous dynamical systems, which are the ones considered in this article, are time-reversible, so in order to create variational integrators that respect this discrete symmetry, it is necessary to consider quadrature formulas that are 'symmetric', which means that the placement of the (interpolation) nodes must be symmetrical with respect to the midpoint of each time interval. This eliminates the use of open Newton–Cotes and Radau integration formulas, for instance. Second, it is difficult to imagine how quadrature rules based on non-polynomial interpolation should be implemented for generic dynamical systems. Numerical integration based on rational functions (see e.g. [8]) either require the location of the poles in advance, or the integration weights cannot be computed explicitly for generic integrands. In the discrete formalism described so far, the former requires the knowledge of contingent singularities of the (discrete) Lagrangian as functions of time, whereas the latter implies that these quadrature rules would have only limited applicability, if at all. Third, numerical integration methods that involve the derivatives of the integrand with respect to the independent variable, that is Turán (see e.g. [9], pp. 42–43) and Birkhoff quadrature formulas (see e.g. [23], Chapter 10), can be used as well, but they call for the time derivatives of the Lagrangian along the (numerical) solutions; it is essentially possible to compute these using either finite differences or automatic differentiation techniques, although that may be difficult and problem-dependent in practice. Furthermore, quadrature rules with arbitrarily high derivatives lead to derivatives of the resultant force, which are usually considered 'unphysical', and thus discarded as options in numerical integration algorithms.

Here we only consider time-independent Lagrangians. Time-dependent dynamical systems can be analysed similarly in the extended phase space formalism [29]. The fully documented library VarInt for Maple 11 and above can be obtained from the author.

Before going into the specifics of each quadrature formula and the herewith associated MAPLE codes, we wish to mention some notational issues. Because only autonomous dynamical systems are considered, it suffices to define the one-step discrete action on the interval $[0, h]$, where $h > 0$ is the time step. Hence, to make the notation somewhat more manageable in MAPLE, we have removed the 'time step' index $k = 0, \ldots, N$ from all variables, as in actual implementations of these variational algorithms the step index is redundant, in the sense that it is translated to a function that returns the updated values of all variables. However, all variables still carry one index, namely the 'time substep' index $i = 0, \ldots, s$. Henceforth we have written the number of nodes $n = s + 1$.

In order to transform any basic quadrature rule to an approximation of the action functional, it is important to notice that the independent variable is time, and that the coordinates and their derivatives with respect to time are approximated by polynomials of order $(n - 1)$. It is possible to design variational integrators based on non-polynomially fitted quadrature rules with the auxiliary module `CreateVarInt`. An example is given at the end of section 4. Nevertheless, the order of *any* variational integrator is determined entirely by the order of the approximation of the discrete action.

3.1. **Newton–Cotes Quadrature.** The closed Newton–Cotes quadrature formulas approximate definite integrals by approximating the integrand $f : \mathbb{R} \to \mathbb{R}$ with an interpolating polynomial evaluated at the node points $x_k = a + kh$, where $k = 0, \ldots, s$, and the stepsize $h = \frac{b-a}{n-1}$:

$$\begin{aligned}\int_a^b f(x)\,\mathrm{d}x &\approx \int_a^b \{\sum_{k=0}^{s} f(x_k)\,\pi_k(x)\}\,\mathrm{d}x \\ &= \sum_{k=0}^{s} f(x_k) \underbrace{\int_a^b \pi_k(x)\,\mathrm{d}x}_{w_k}.\end{aligned}$$

Here, $\{\pi_k(x)\}$ is a polynomial basis, and w_k are known as the weights; these weights are usually calculated by integration of Lagrange polynomials, although one is in principle free to select any polynomial basis for the interpolation.

3.2. **Romberg Quadrature.** Another family of classical integration formulas with equidistant nodes is the one due to Romberg. Romberg quadrature distinguishes itself from Newton–Cotes quadrature in that it always uses the same basic two-point approximation, the composite trapezium rule, yet recursively by inserting nodes at the centres of all (sub)intervals. The essence of Romberg quadrature is that a Richardson extrapolation procedure is applied to the composite trapezium rule to obtain higher-order approximations to the integral under evaluation.

It is important to note that the composite trapezium rule leads to a continuous approximation of the integrand, yet its derivative with respect to the independent variable is discontinuous at each node. Hence, the naive implementation of Romberg quadrature seems impossible to generate variational integrators, as we require that $q \in \mathcal{C}^1([t_k, t_{k+1}], \mathbb{R})$ for $k = 0, \ldots, N - 1$. Nevertheless, we can still use a 'modified' trapezium rule and Richardson extrapolation in conjunction with a sufficiently smooth interpolating function, at the cost of losing the adaptivity of the algorithm. Again, the composite trapezium rule is used as a basic approximation,

though now the interpolating function is not piecewise linear but rather it is chosen such that both q and $\dot{q}$ are well-defined at each node.

3.3. **Gaussian Quadrature.** A class of n-point quadrature rules that integrate up to $(2n-1)$st-degree polynomials exactly are the Gaussian ones by evaluating a weighted sum of function values. The integrand is assumed to be sufficiently smooth, specifically it is a $\mathcal{C}^{2n}([-1,1],\mathbb{R})$ function.

$$\int_a^b f(x)\,\omega(x)\,\mathrm{d}x = \sum_{k=1}^{n} w_k f(x_k) + R_n, \tag{13}$$

where the 'optimal' values for the weights w_k depend on the placement of the nodes x_k along the interval $[a,b]$. R_n denotes the remainder for a Gaussian integration formula with n nodes,

$$R_n = \frac{f^{(2n)}(\xi)}{(2n)!}\int_a^b \omega(x)\,\phi_n^2(x)\,\mathrm{d}x,$$

where $a<\xi<b$, and $\phi_n(x)$ is the related nth degree orthogonal polynomial (see [28], pp. 180–181). As usual, $\omega(x)$ denotes a positive weight function appearing in the integrand. In the case of interest for variational integrators, the nodes are placed symmetrically over a finite interval, for which $[-1,1]$ is commonly used. For an integral over a arbitrary but finite interval $[a,b]$ the linear transformation $x \mapsto \frac{1}{2}(b-a)\,x + \frac{1}{2}(a+b)$ can then be used. An overview of the various quadrature formulas of the Gauss family can be found in the chapter on numerical analysis in the book by Abramowitz and Stegun [1], for example. Here, we shall discuss the quadrature rules based on the Legendre, Chebyshev and Lobatto nodes. The nodes for the Gauss–Radau quadrature formulas are not distributed symmetrically across the interval of integration, so that they cannot be used for the design of variational integrators for autonomous dynamical systems.

3.3.1. *Gauss–Legendre Quadrature.* In Gauss–Legendre quadrature formulas the weight function $\omega(x)=1$, which is known as the Legendre weight function. The nodes x_k with $k=1,\ldots,n$ for the n-point Gauss–Legendre quadrature formulas are the zeros of the Legendre polynomials $P_n(x)$. The corresponding weights are given by

$$w_k = \frac{2}{1-x_k^2}\frac{1}{\left[P_n'(x_k)\right]^2}, \tag{14}$$

where the prime indicates the derivative with respect to the argument. It is important to note that the zeros of the Legendre polynomials come in pairs, so that the quadrature rule is symmetric about the origin. Furthermore, the zeros lie in the interval $(-1,1)$, that is, they do not include the endpoints.

The fact that the endpoints of the integration interval do not appear explicitly in the quadrature formula means that it is necessary to 'include' the endpoints by means of extrapolation; the values of the coordinates and their derivatives are indeed specified at one such a point for initial-value problems. The idea is to interpolate the coordinates with an $(n-1)$st degree polynomial through the interior points $(i=1,\ldots,s-1)$, as before, and extrapolate to the endpoints of the integration interval ($i=0$ and $i=s$). It is then possible to express the first ($i=1$) and last ($i=s-1$) of the interior points in terms of the remaining interior points and the endpoints. In that way, the endpoints can be included in accordance with the

quadrature nodes. Although polynomial extrapolation is notorious for being very inaccurate outside the interval of the interpolation, we shall assume that the time step h is sufficiently small to overcome the issues associated herewith.

3.3.2. *Gauss–Chebyshev Quadrature.* The n-point Gauss–Chebyshev integration formulas come in two slightly different flavours. The first type of Gauss–Chebyshev quadrature rule has nodes at the roots of the Chebyshev polynomials of the first kind, $T_n(x)$. These are

$$x_k^{(1)} = \cos\theta_k^{(1)}, \qquad \theta_k^{(1)} = \frac{2k-1}{n}\frac{\pi}{2}, \tag{15}$$

and the corresponding weights are

$$w_k^{(1)} = \frac{\pi}{n}. \tag{16}$$

The related quadrature formula reads

$$\int_{-1}^{-1} f(x)\,\mathrm{d}x \approx \sum_{k=1}^{n} w_k^{(1)} f\left(x_k^{(1)}\right)\sqrt{1-\left(x_k^{(1)}\right)^2}, \tag{17}$$

where the square-root is the weight function, now appearing on the right-hand side. Similarly, the integration formula for second type of Gauss–Chebyshev integration formulas is

$$\int_{-1}^{-1} f(x)\,\mathrm{d}x \approx \sum_{k=1}^{n} w_k^{(2)} \frac{f\left(x_k^{(2)}\right)}{\sqrt{1-\left(x_k^{(2)}\right)^2}}, \tag{18}$$

where now the nodes are given by

$$x_k^{(2)} = \cos\theta_k^{(2)}, \qquad \theta_k^{(2)} = \frac{k}{n+1}\pi, \tag{19}$$

which are the zero loci of the Chebyshev polynomials of the second kind, $U_n(x)$. The matching weights are

$$w_k^{(2)} = \frac{\pi}{n+1}\sin^2\theta_k^{(2)}. \tag{20}$$

The Gauss–Chebyshev quadrature formulas are especially suited for integrands with factors of $\sqrt{1-x^2}$ either in their numerators or denominators.

Please observe that the extrapolation to the endpoints of the integration interval does not yield any singularities due to the weight function. The coordinates and velocities are extrapolated and these extrapolations are substituted, so that the sum is indeed evaluated at the correct nodes.

3.3.3. *Fejér Quadrature.* The Gauss–Chebyshev quadratures discussed in the previous paragraph were defined with respect to non-trivial weight functions $\omega(x) = 1/\sqrt{1-x^2}$ and $\omega(x) = \sqrt{1-x^2}$ for the integration formulas based on the Chebyshev polynomials of the first and second kind, respectively. As for all Gauss quadratures rules, these can be defined relative to different weight functions. For $\omega(x) = 1$ one obtains the formulas due to Fejér [7]. The nodes for the integration rules based

on the Chebyshev polynomials of the first and second kind are identical to equations (15) and (19), respectively. However, the weights are now

$$w_k^{(1)} = \frac{2}{n}\left[1 - 2\sum_{j=1}^{\lfloor n/2\rfloor} \frac{\cos\left(2j\theta_k^{(1)}\right)}{4j^2-1}\right], \tag{21}$$

and

$$w_k^{(2)} = \frac{4\sin\theta_k^{(2)}}{n+1} \sum_{j=1}^{\lfloor (n+1)/2\rfloor} \frac{\sin\left((2j-1)\,\theta_k^{(2)}\right)}{2j-1}, \tag{22}$$

where $\theta_k^{(1)}$ and $\theta_k^{(2)}$ are as before.

Alternatively, the zeros of the nth-degree Chebyshev polynomial of the third kind $V_n(x)$ can be used,

$$x_k^{(3)} = \cos\theta_k^{(3)}, \qquad \theta_k^{(3)} = \frac{2k-1}{2n+1}\pi, \tag{23}$$

as well as those of the nth-degree Chebyshev polynomial of the fourth kind $W_n(x)$,

$$x_k^{(4)} = \cos\theta_k^{(4)}, \qquad \theta_k^{(4)} = \frac{2k}{2n+1}\pi. \tag{24}$$

The corresponding weights are

$$w_k^{(3)} = \frac{4\sin\theta_k^{(3)}}{n+\frac{1}{2}} \sum_{j=1}^{\lfloor (n+1)/2\rfloor} \frac{\sin\left((2j-1)\,\theta_k^{(3)}\right)}{2j-1}, \tag{25}$$

and

$$w_k^{(4)} = \frac{4\sin\theta_k^{(4)}}{n+\frac{1}{2}} \sum_{j=1}^{\lfloor (n+1)/2\rfloor} \frac{\sin\left((2j-1)\,\theta_k^{(4)}\right)}{2j-1}, \tag{26}$$

respectively, as shown by [26].

Related to Fejér quadrature formulas is the one by [5], which is nothing but Fejér's second rule with the nodes -1 and 1 added. Define

$$\theta_k = \frac{k-1}{n-1}\pi, \qquad k = 1,\ldots,n.$$

The Clenshaw–Curtis nodes are then simply $x_k = \cos\theta_k$, and the associated weights are given by

$$w_k = \frac{c_k}{n}\left[1 - 2\sum_{j=1}^{\lfloor (n+1)/2\rfloor}{}^{*} \frac{\cos 2j\theta_k}{4j^2-1}\right], \tag{27}$$

where $c_k = 2 - \delta_{0,k \bmod n}$, and $\sum{}^{*}$ signifies that the last term in the sum should be halved.

3.3.4. *Gauss–Lobatto Quadrature.* Additional Gaussian integration rules that include both endpoints are the Gauss–Lobatto ones:

$$\int_{-1}^{-1} f(x)\,\mathrm{d}x \approx \frac{2}{n(n-1)}\left[f(-1)+f(1)\right]+\sum_{k=2}^{n-1} w_k f(x_k). \tag{28}$$

The interior nodes are the zeros of the derivative of the Legendre polynomials, that is they satisfy $P'_{n-1}(x)=0$, and the interior weights can be calculated to be

$$w_k = \frac{2}{n(n-1)}\frac{1}{\left[P_{n-1}(x_k)\right]^2}. \tag{29}$$

3.4. **Chebyshev Quadrature.** Somewhat related to the quadrature formulas of the Gaussian type is the equal-weight integration formula by Chebyshev:

$$\int_{-1}^{-1} f(x)\,\mathrm{d}x \approx \frac{2}{n}\sum_{k=1}^{n} f(x_k). \tag{30}$$

The nodes are the solutions to the equation $G_n(x)=0$, where $G_n(x)$ is the polynomial part of [14]

$$F_n(x) = x^n \exp\left[\frac{n}{2}\int_{-1}^{1}\ln\left(1-\frac{t}{x}\right)\mathrm{d}t\right]. \tag{31}$$

The integral inside the exponential can be calculated easily,

$$\int_{-1}^{1}\ln\left(1-\frac{t}{x}\right)\mathrm{d}t = -2+(1+x)\ln\left(1+\frac{1}{x}\right)+(1-x)\ln\left(1-\frac{1}{x}\right).$$

The zeros of $G_n(x)$ are known to be real only for $n\le 7$ and $n=9$. Hence, the use of Chebyshev quadrature is restricted to these values.

3.5. **Takahasi–Mori Quadrature.** For the numerical computation of integrals over infinite intervals $(-\infty,\infty)$ the composite trapezium rule is noted for its excellent results in terms of accuracy and efficiency compared to quadrature formulas with the same density of sampling points [30], that is, for any analytical function g that vanishes at infinity,

$$\int_{-\infty}^{\infty} g(x)\,\mathrm{d}x \approx \eta\sum_{k=-\infty}^{\infty} g(k\eta),$$

where in practice the infinite sum itself converges often quite rapidly. We can take advantage of the performance of the trapezium rule by applying a variable transformation, $x\mapsto\varphi(t)$, to integrals over finite intervals:

$$\begin{aligned}\int_{-1}^{1} f(x)\,\mathrm{d}x &= \int_{-\infty}^{\infty} f(\varphi(t))\,\varphi'(t)\,\mathrm{d}t\\ &\approx \eta\sum_{k=-\infty}^{\infty} f(k\eta)\,\varphi'(k\eta).\end{aligned}$$

The method proposed by Schwartz [27] involves the transformation $\varphi(t)=\tanh t$, for which the resulting quadrature formula has an asymptotic error of $\mathcal{O}\left(\exp\left(-c\sqrt{N}\right)\right)$ with $\eta=\pi/\sqrt{N}$ [12], where N denotes the number of function evaluations, and $c\in\mathbb{R}$ depends on the integrand and the particular variable

transformation. For $\varphi(t) = \operatorname{erf} t$ the error is $\mathcal{O}\left(\exp\left(-c\sqrt[3]{N^2}\right)\right)$ asymptotically (see e.g. [25] for more details on these and other variable transformations). In fact, for all functions $f \in H^p(D)$, $1 < p \leq \infty$, the Hardy spaces on the unit disc $D = \{z \in \mathbb{C} \mid |z| < 1\}$, Andersson [2] has shown that the bound on the asymptotic error of any quadrature formula is $\mathcal{O}\left(N^{1-1/(2p)} \exp\left(-c\sqrt{N}\right)\right)$.

Double exponential quadrature formula dates back to the work by Takahasi and Mori [31], who improved on the transformation method by Schwartz [27]. Their integration rule accelerates the convergence of one-dimensional integrals by introducing a suitable variable transformation that result in double exponential decay of the integrand: $\varphi(t) = \tanh\left(\frac{\pi}{2}\sinh t\right)$. Rather than looking at functions that belong to the Hardy classes $H^p(D)$ with $p > 1$, we can focus on the more modest class of integrable functions over $(-1, 1)$, possibly with algebraic or logarithmic singularities at the endpoints ± 1, and a finite number of singularities outside the interval of integration. Then, the asymptotic error of the quadrature formula behaves as $\mathcal{O}(\exp(-cN/\ln N))$; the constant c is related to the location of the singularities of the integrand after the application of the variable transformation. The optimal value of

$$\eta = \frac{2}{N} \ln 2\Delta N,$$

where Δ is the distance between the real axis and the nearest singularity of the integrand after the variable transformation has been applied; the transformed integrand is thus regular in the strip $|\Im(z)| < \Delta$. In case the original function $f(z)$, $z \in \mathbb{C}$, only has a singularity at $z = \infty$, we easily compute that $\Delta = \frac{\pi}{2}$. At the optimal step η, the nodes in the interval $(-1, 1)$ tend to cluster near the boundaries, especially for small N.

The Takahasi–Mori, or tanh-sinh, formula,

$$\int_{-1}^{1} f(x)\,\mathrm{d}x \approx \eta \frac{\pi}{2} \sum_{k=-n}^{n} f\left(\tanh\left(\frac{\pi}{2}\sinh k\eta\right)\right) \frac{\cosh k\eta}{\cosh^2\left(\frac{\pi}{2}\sinh k\eta\right)}, \tag{32}$$

has been shown to be fast and accurate in high-precision experimental mathematics [3]; in practice we often choose η adaptively. Recently, Borwein and Ye [4] have shown that the Takahasi–Mori quadrature formula converges quadratically for all integrands $f \in H^2(D)$ in the limit of $N \to \infty$.

All these transformed quadrature formulas based on the trapezium rule have exponential decay of the asymptotic error, which basically means that halving the stepsize roughly doubles the number of correct digits. Note, however, that the quadrature formulas are not exact for polynomials, in contrast to the Gaussian quadrature formulas.

4. Examples

The symplectic partitioned Runge–Kutta methods form a well-known class of variational integrators for conservative dynamical systems. For non-conservative systems probably the best studied example is the symplectic Newmark algorithm, as described in [18]. Beyond these the number of variational integrators is limited, mainly because the manual effort to generate these (higher-order) variational integrators is substantial.

VarInt is a library that enables anyone with a Maple distribution to create and analyse new variational integrators with ease. The module VarInt has four main

procedures: `VarInt`, `CreateVarInt`, `ExtractAlgorithm`, and `IntegrateSystem`, which provides basic functionality for the numerical analysis of one-dimensional problems. `VarInt` computes the (forced) discrete Euler–Lagrange equations. In order to obtain an actual recipe that allows us to compute the discrete flow efficiently, we have to manipulate the expressions returned by `VarInt`, which depends highly on the functional form of the Lagrangian, and is hence best done interactively. For separable Lagrangians

$$L(q,\dot{q}) = T(\dot{q}) - V(q), \tag{33}$$

with $T\colon \mathbf{T}\mathbb{Q} \to \mathbb{R}$ a quadratic kinetic energy function and $V\colon \mathbb{Q} \to \mathbb{R}$ the potential energy, an ancillary procedure `ExtractAlgorithm` is included in VarInt. It aids in the extraction of such a one-step map, even for dynamical systems with generic non-conservative forces. As such, it greatly enhances the potential development variational integrators for non-conservative forces up to arbitrary order, which has only been touched upon scantily thus far.

The module `CreateVarInt` is similar in design as `VarInt` with the significant difference that the approximation to the discrete action can be supplied manually by specifying the nodes, weights and weight function of the numerical integration formula, and the interpolation procedure, which is polynomial by default. `CreateVarInt` therefore extends the `VarInt` by allowing new quadrature rules to be defined and the creation of non-polynomially fitted variational integration algorithms.

To see the full scope of VarInt, we shall first look at simple problems. Consider a two-point Newton–Cotes approximation of the action and the (non-conservative) Rayleigh force. Then, we can obtain the discrete Euler–Lagrange equations with `VarInt` as follows:

```
> restart;                                    #clear memory
> with(VarInt):                               #load VarInt
> dEL1:=VarInt(2,L,F,NewtonCotes,p,q,h);      #obtain dEL equations.
```

Since, we have not (yet) specified the functional forms of the Lagrangian and the Rayleigh force, we the expressions MAPLE returns are fully implicit. To obtain a more applicable representation of the two-point variational Newton–Cotes integrator, we define a separable Lagrangian (33), and extract the algorithm:

```
> L:=(q,Dq)->1/2*M*Dq^2-V(q):                 #define Lagrangian
> dEL2:=VarInt(2,L,F,NewtonCotes,p,q,h):      #obtain dEL equations
> ExtractAlgorithm(dEL2,p,q,V,F);             #obtain algorithm.
```

Here, M is the mass; in the case of vectorial coordinates and momenta, `M` has to be interpreted as the mass matrix. The one-step map $(q_0,p_0) \mapsto (q_1,p_1)$ reads

$$q_1 = q_0 + h\frac{p_0}{M} - \frac{h^2}{2M}\left[\nabla V(q_0) - F\left(q_0, \frac{q_1-q_0}{h}\right)\right], \tag{34a}$$

$$p_1 = p_0 - \frac{h}{2}\left[\nabla V(q_0) + \nabla V(q_1) - F\left(q_0, \frac{q_1-q_0}{h}\right) - F\left(q_1, \frac{q_1-q_0}{h}\right)\right], \tag{34b}$$

The algorithm is implicit for generic `F`. For conservative dynamical systems the algorithm reduces to the the famous second-order Störmer–Verlet algorithm, which

is sometimes referred to as the leapfrog:

$$q_1 = q_0 + h\frac{p_0}{M} - \frac{h^2}{2M}\nabla V\left(q_0\right), \tag{35a}$$

$$p_1 = p_0 - \frac{h}{2}\left[\nabla V\left(q_0\right) + \nabla V\left(q_1\right)\right], \tag{35b}$$

and is fully explicit. It can be obtained in the active MAPLE worksheet in several ways:

```
> eval(%,F=0);                                  #alternative 1
> F:=(q,Dq)->0:                                 #alternative 2
> dEL3:=VarInt(2,L,F,NewtonCotes,p,q,h);        #alternative 2 (cont'd)
> ExtractAlgorithm(dEL3,p,q,V,F);               #alternative 2 (cont'd)
> dEL4:=VarInt(2,L,0,NewtonCotes,p,q,h);        #alternative 3
> ExtractAlgorithm(dEL3,p,q,V,F);               #alternative 3 (cont'd).
```

As it happens, the Newton–Cotes, Romberg, Gauss–Lobatto and Clenshaw–Curtis quadrature rules with two nodes are identical, so that their variational integrators are the same.

Incidentally, for three nodes the Newton–Cotes, Gauss–Lobatto and Clenshaw–Curtis quadrature (Simpson's) formulas coincide. The MAPLE code

```
> dEL5:=VarInt(3,L,0,GaussLobatto,p,q,h);       #obtain dEL equations
> ExtractAlgorithm(dEL5,p,q,V,F);               #obtain algorithm
```

results in the fourth-order algorithm for conservative dynamical systems reported in [6],

$$q_1 = q_0 + \frac{h}{2}\frac{p_0}{M} - \frac{h^2}{24M}\left[2\nabla V\left(q_0\right) + \nabla V\left(q_1\right)\right], \tag{36a}$$

$$q_2 = q_0 + h\frac{p_0}{M} - \frac{h^2}{6M}\left[\nabla V\left(q_0\right) + 2\nabla V\left(q_1\right)\right], \tag{36b}$$

$$p_2 = p_0 - \frac{h}{6}\left[\nabla V\left(q_0\right) + 4\nabla V\left(q_1\right) + \nabla V\left(q_2\right)\right]. \tag{36c}$$

Equation (36a) has to be solved iteratively for generic (non-linear) potentials. Equations (36b)–(36c) are clearly explicit.

For four nodes these three families of quadrature rules lead to different variational integrators. The variational Newton–Cotes integrator, which is based on Simpson's $\frac{3}{8}$ rule, is easily found to be

$$q_1 = q_0 + \frac{h}{3}\frac{p_0}{M} - \frac{h^2}{648M}\left[27\nabla V\left(q_0\right) + 14\nabla V\left(q_1\right) - 5\nabla V\left(q_2\right)\right], \tag{37a}$$

$$q_2 = q_0 + \frac{2h}{3}\frac{p_0}{M} - \frac{h^2}{324M}\left[27\nabla V\left(q_0\right) + 38\nabla V\left(q_1\right) + 7\nabla V\left(q_2\right)\right], \tag{37b}$$

$$q_3 = q_0 + h\frac{p_0}{M} - \frac{h^2}{8M}\left[\nabla V\left(q_0\right) + 2\nabla V\left(q_1\right) + \nabla V\left(q_2\right)\right], \tag{37c}$$

$$p_3 = p_0 - \frac{h}{8}\left[\nabla V\left(q_0\right) + 3\nabla V\left(q_1\right) + 3\nabla V\left(q_2\right) + \nabla V\left(q_3\right)\right]. \tag{37d}$$

Similarly, the variational Clenshaw–Curtis integrator with four nodes is

$$q_1 = q_0 + \frac{3h}{14}\frac{p_0}{M} - \frac{h^2}{13440M}\left[320\nabla V(q_0) + 259\nabla V(q_1) - 21\nabla V(q_2)\right], \tag{38a}$$

$$q_2 = q_0 + \frac{6h}{7}\frac{p_0}{M} - \frac{h^2}{13440M}\left[1280\nabla V(q_0) + 3339\nabla V(q_1) + 259\nabla V(q_2)\right], \tag{38b}$$

$$q_3 = q_0 + \frac{15h}{14}\frac{p_0}{M} - \frac{h^2}{84M}\left[10\nabla V(q_0) + 28\nabla V(q_1) + 7\nabla V(q_2)\right], \tag{38c}$$

$$p_3 = p_0 - \frac{h}{18}\left[2\nabla V(q_0) + 7\nabla V(q_1) + 7\nabla V(q_2) + 2\nabla V(q_3)\right]. \tag{38d}$$

The variational integrators that derive from the Gauss–Lobatto quadrature rules correspond to the well-known Lobatto IIIA/IIIB algorithms, and their forms can be found in the literature.

All variational Gauss–Legendre integrators have been shown to be equal to the Gauss collocation methods. As an example, the Gauss–Legendre variational integrator with two nodes is easily found to be

$$q_1 = q_0 + h\frac{p_0}{m} - \frac{h^2}{12m}\left[c_-\nabla V\left(q_+\right) + c_+\nabla V\left(q_-\right)\right], \tag{39}$$

$$p_1 = p_0 - \frac{h}{2}\left[\nabla V\left(q_+\right) + \nabla V\left(q_-\right)\right], \tag{40}$$

where we have defined $q_\pm = \frac{1}{2}\left(q_0 + q_1\right) \pm \frac{1}{6}\sqrt{3}\left(q_0 - q_1\right)$, and $c_\pm = 3 \pm \sqrt{3}$. The Gauss–Legendre and the Chebyshev quadrature formulas with two nodes happen to coincide, so that their variational integrators are identical (39). More details and examples can be found in the help pages, which can be accessed by executing one of the following commands:

```
> ?VarInt                                  #package overview
> ?VarInt[VarInt]                          #help page
> ?VarInt[ExtractAlgorithm]                #help page
> ?VarInt[CreateVarInt]                    #help page
> ?VarInt[IntegrateSystem]                 #help page.
```

Finally, we shall take a look at the `CreateVarInt` module. The syntax is slightly different from `VarInt`, as one can see below:

```
> Digits:=16:                              #numerical precision
> x:=0.5904158239150231:                   #positive node
> w:=0.9964248649058515:                   #weight
> etc:=1,L,0,p,q,h:                        #shorthand
> CreateVarInt(-1..1,[-x,x],[w,w],etc):    #obtain dEL
> ExtractAlgorithm(%,p,q,V);               #obtain algorithm
```

The first argument is the range on which the nodes are defined, so that the nodes, supplied as a list as the second argument to `CreateVarInt`, can be transformed appropriately. The third argument is the list of weights associated with these nodes. The fourth argument is the weight function, which in this case is the unit function. The fifth through to the ninth argument are the Lagrangian function, the Rayleigh function, and the labels for the canonical momenta, canonical coordinates

and the time step, respectively. The tenth argument is optional, and it is not shown here; it takes the handle of an interpolation procedure, which must have the same syntax as the built-in procedures for data interpolation, as specified in the documentation of the `CurveFitting` package. If the tenth argument is omitted, the standard polynomial interpolation procedure `PolynomialInterpolation` (also known as `interp`) is used internally. As an example, consider a custom yet naive implementation of polynomial interpolation:

```
> Poly:=proc(xdata,ydata,z)                    #custom interpolation
    local c,n,Eqs,Var,Fun;
    n:=nops(xdata):
    Fun:=x->add(c[k]*x^(k-1),k=1..n):
    Eqs:={seq(Fun(xdata[m])=ydata[m],m=1..n)}:
    Var:={seq(c[m],m=1..n)}:
    assign(solve(Eqs,Var)):
    collect(factor(Fun(z)),z);
  end proc:
> etc:=1,L,0,p,q,h,Poly;                        #shorthand
> CreateVarInt(-1..1,[-1,1],[1,1],etc):         #obtain dEL
> ExtractAlgorithm(%,p,q,V);                    #obtain algorithm.
```

The code obviously yields the Störmer–Verlet algorithm (35).

The values for the nodes and weights shown are such that the underlying quadrature rule integrates any linear combinations of the set $\{e^{\pm\nu x}, x\,e^{\pm\nu x}\}$ with $\nu = 1$ exactly on the interval $[-1, 1]$. Recently, non-polynomially fitted quadrature rules have moved increasingly to the centre of attention [16, 35, 33], especially exponentially fitted ones for numerical integration algorithms for ordinary differential equations (see [34] and references therein for more details). The idea behind is to translate the philosophy behind Gaussian integration formulas, that is that they integrate polynomials exactly, to non-polynomial functions, in particular exponentials and trigonometric functions, based on the formalism developed by Ixaru [15]. That leads to a set non-linear conditions, from which the nodes and weights can be computed (numerically). Unfortunately, the nodes and corresponding weights for these exponentially fitted quadrature rules are not determined uniquely.

The optional argument enables us to provide alternative interpolation routines, which can be practical both as a diagnostic tool and as a interface to create new variational integrators that are designed for specific dynamical systems. Quadrature rules based on rational interpolation [32], for instance, might be of use in the simulations of dynamical systems with singularities, such as N-body problems in astrophysics and molecular dynamics for instance.

5. Conclusion

It is a well-established fact that simulations of (non-linear) dynamical systems, both with non-conservative forces and without, benefit greatly from the preservation of their geometric structures, especially over long time spans as compared to the characteristic time scales of the systems at hand. Variational integrators, and more generally geometric numerical integrators, are ideally suited for such simulations. The discrete variational formalism is both mathematically natural and

computationally practical. We have demonstrated that one can explore and design variational integrators systematically with a computer algebra system, such as MAPLE. Some of these variational integrators correspond to well-known classes of geometric numerical algorithms, such as the symplectic partitioned Runge–Kutta methods. However, few variational integrators have been reported that lie outside of the standard classification, although the discrete variational formalism is certainly not restricted to it. With the procedures we have presented to compute variational integrators based on different approximations of the action functional one can venture beyond the geometric numerical algorithms one usually encounters. The discrete flow maps one obtains can be either general, and serve as templates for generic problems, or optimized for a specific problem thanks to the symbolic capabilities of a computer algebra system.

REFERENCES

1. M. Abramowitz and I.A. Stegun (eds.), *Handbook of Mathematical Functions*, Dover Publications, Mineola, NY, 1972.
2. J.E. Andersson, *Optimal Quadrature of H^p Functions*, Mathematische Zeitschrift **172** (1980), 55–62.
3. D.H. Bailey, K. Jeyabalan, and X.S. Li, *A Comparison of Three High-Precision Quadrature Schemes*, Experimental Mathematics **14** (2005), no. 3, 317–329.
4. J.M. Borwein and L. Ye, *Quadratic Convergence of the* tanh-sinh *Quadrature Rule*, D-Drive Preprint #342, 2006.
5. C.W. Clenshaw and A.R. Curtis, *A Method for Numerical Integration on an Automatic Computer*, Numerische Mathematik **2** (1960), 197–205.
6. W.M. Farr and E. Bertschinger, *Variational Integrators for the Gravitational N-Body Problem*, The Astrophysical Journal **663** (2007), no. 2, 1420–1433.
7. L. Fejér, *Mechanische Quadraturen mit positiven Cotesschen Zahlen*, Mathematische Zeitschrift **37** (1933), 287–309.
8. W. Gautschi, *The Use of Rational Functions in Numerical Quadrature*, Journal of Computational and Applied Mathematics **133** (2001), no. 1-2, 111–126.
9. ———, *Orthogonal Polynomials and Special Functions. Computation and Applications*, Lecture Notes in Mathematics, vol. 1883, Springer, Berlin, Heidelberg, 2006.
10. Z. Ge and J.E. Marsden, *Lie–Poisson Hamilton–Jacobi Theory and Lie–Poisson Integrators*, Physics Letters A **133** (1988), no. 3, 134–139.
11. A. Griewank and A. Walther, *Evaluating Derivatives. Principles and Techniques of Algorithmic Differentiation*, Applied Mathematics, vol. 105, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2008.
12. S. Haber, *The* tanh *Rule for Numerical Integration*, SIAM Journal on Numerical Analysis **14** (1977), no. 4, 668–685.
13. E. Hairer, C. Lubich, and G. Wanner, *Geometric Numerical Integration. Structure-Preserving Algorithms for Ordinary Differential Equations*, Springer Series in Computational Mathematics, vol. 31, Springer, Berlin, Heidelberg, New York, 2006.
14. F.B. Hildebrand, *Introduction to Numerical Analysis*, Dover Publications, Mineola, NY, 1987.
15. L.G. Ixaru, *Operations on Oscillatory Functions*, Computer Physics Communications **105** (1997), no. 1, 1–19.
16. L.G. Ixaru and B. Paternoster, *A Gauss Quadrature Rule for Oscillatory Integrands*, Computer Physics Communications **133** (2001), no. 2-3, 177–188.
17. O. Junge, J.E. Marsden, and S. Ober Blöbaum, *Discrete Mechanics and Optimal Control*, Proceedings of the 16th IFAC Conference on Decision and Control, 2005.
18. C. Kane, J.E. Marsden, M. Ortiz, and M. West, *Variational Integrators and the Newmark Algorithm for Conservative and Dissipative Mechanical Systems*, International Journal for Numerical Methods in Engineering **49** (2000), no. 10, 1295–1325.
19. L. Kharevych, Y. Weiwei, Y. Tong, E. Kanso, J.E. Marsden, P. Schröder, and M. Desbrun, *Geometric, Variational Integrators for Computer Animation*, Proceedings of the 2006 ACM

SIGGRAPH/Eurographics Symposium on Computer Animation (Aire-la-Ville, Switzerland), Eurographics Association, 2006, pp. 43–51.
20. B. Leimkuhler and S. Reich, *Simulating Hamiltonian Dynamics*, Cambridge Monographs on Applied and Computational Mathematics, vol. 14, Cambridge University Press, Cambridge, UK, 2005.
21. A. Lew, J.E. Marsden, M. Ortiz, and M. West, *Asynchronous Variational Integrators*, Archive for Rational Mechanics and Analysis **167** (2003), no. 2, 85–146.
22. ———, *Variational Time Integrators*, International Journal for Numerical Methods in Engineering **60** (2004), no. 1, 153–212.
23. G.G. Lorentz, K. Jetter, and S.D. Riemenschneider, *Birkhoff Interpolation*, Encyclopedia of Mathematics and Its Applications, vol. 19, Cambridge University Press, Cambridge, UK, 1984.
24. J.E. Marsden and M. West, *Discrete Mechanics and Variational Integrators*, Acta Numerica **10** (2001), 357–514.
25. M. Mori, *Quadrature Formulas Obtained by Variable Transformation and the DE-Rule*, Journal of Computational and Applied Mathematics **12-13** (1985), 119–130.
26. S.E. Notaris, *Interpolatory Quadrature Formulae with Chebyshev Abscissae of the Third or Fourth Kind*, Journal of Computational and Applied Mathematics **81** (1997), no. 1, 83–99.
27. C. Schwartz, *Numerical Integration of Analytic Functions*, Journal of Computational Physics **4** (1969), no. 1, 19–29.
28. J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, Texts in Applied Mathematics, vol. 12, Springer, New York, NY, 2002.
29. J. Struckmeier, *Hamiltonian Dynamics on the Symplectic Extended Phase Space for Autonomous and Non-Autonomous Systems*, Journal of Physics A: Mathematical and General **38** (2005), no. 6, 1257–1278.
30. H. Takahasi and M. Mori, *Error Estimation in the Numerical Integration of Analytic Functions*, Report of the Computer Centre, University of Tokyo **3** (1970), 41–108.
31. ———, *Double Exponential Formulas for Numerical Integration*, Publications of the Research Institute for Mathematical Sciences **9** (1974), no. 3, 721–741.
32. W. Van Assche and I. Vanherwegen, *Quadrature Formulas Based on Rational Interpolation*, Mathematics of Computation **61** (1993), no. 204, 765–783.
33. G. Vanden Berghe and M. Van Daele, *Trigonometric Polynomial or Exponential Fitting Approach?*, Journal of Computational and Applied Mathematics **233** (2009), no. 4, 969–979.
34. ———, *Symplectic Exponentially-Fitted Four-Stage Runge–Kutta Methods of the Gauss Type*, Numerical Algorithms (2010), in press.
35. G. Vanden Berghe, M. Van Daele, and H. Vande Vyver, *Exponentially Fitted Quadrature Rules of The Gauss Type for Oscillatory Integrands*, Applied Numerical Mathematics **53** (2005), no. 2-4, 509–526.

Department of Physics and Astronomy, University of Turku, Finland
E-mail address: christian.hellstrom@utu.fi