J. Nonlinear Sci. Vol. 3: pp. 1–33 (1993)

Journal of
Nonlinear
Science

Numerical Integration of Ordinary Differential Equations on Manifolds

P. E. Crouch[1] and R. Grossman[2]

[1] Center for Systems Science and Engineering, Arizona State University, Tempe, AZ 85287-7606, USA

[2] Laboratory for Advanced Computing, University of Illinois at Chicago, Chicago, IL 60680, USA

Received August 15, 1991; accepted for publication July 13, 1992
Communicated by Paul Channell

Summary. This paper is concerned with the problem of developing numerical integration algorithms for differential equations that, when viewed as equations in some Euclidean space, naturally evolve on some embedded submanifold. It is desired to construct algorithms whose iterates also evolve on the same manifold. These algorithms can therefore be viewed as integrating ordinary differential equations on manifolds. The basic method "decouples" the computation of flows on the submanifold from the numerical integration process. It is shown that two classes of single-step and multistep algorithms can be posed and analyzed theoretically, using the concept of "freezing" the coefficients of differential operators obtained from the defining vector field. Explicit third-order algorithms are derived, with additional equations augmenting those of their classical counterparts, obtained from "obstructions" defined by nonvanishing Lie brackets.

Key words. numerical integration, manifold, differential equation flow, lie algebra, algorithm, symbolic computation, frozen coefficients

AMS Subject Classifications. 34A50, 34A34, 65L06, 93C15, 58F99

1. Introduction

1.1. Numerical Algorithms and Constraints

The flow of many systems of ordinary differential equations preserves certain constraints. For example, a conservative mechanical system preserves energy, a rotating

rigid body in space preserves angular momentum, a Hamiltonian system of equations preserves the symplectic structure, and the configuration variables of a spherical pendulum are confined to a sphere. As these examples illustrate, the dynamical variables may be constrained by a conservation law, a manifold or a group structure. More generally, there may be some mixed algebraic-differential equation constraining the dynamic variables. A general-purpose numerical algorithm usually does not preserve these constraints, although there is a large variety of algorithms that do. Without trying to be complete we mention the following:

Perhaps the oldest algorithms of this type are those that preserve energy, see Chorin, Hughes, Marsden, and McCracken [6] and the references contained there for examples and further discussion of algorithms of this type. Algorithms that preserve symplectic invariants are also important and quite old. Work in this area began with de Vogelaere [11] several decades ago. Important contributions have been made by Deprit [12], Dragt and Finn [13], Feng [14], and Ruth [25]. Recently, there has been a resurgence of interest in this area, sparked by the paper of Channell and Scovel [5]. The paper by Scovel [27] contains a recent survey of this work.

Several years ago, motivated by the work of Channell and Scovel on symplectic integrators, Zhong and Marsden [16] introduced a class of Poisson integrators, that is, a class of integrators that preserve the Poisson structure of a system. A selection of other recent works on symplectic and Poisson integrators includes [1,4,30] Oftentimes familiar algorithms automatically preserve more invariants. For example, Sanz-Serna has investigated Runge-Kutta algorithms that are also symplectic [26].

Beginning with the work of Gear [15], a number of authors have developed numerical algorithms that integrate general differential-algebraic systems, (see [23,24] and the monograph [18]).

Our interest here is to introduce a class of numerical algorithms that naturally evolve on a constraint manifold. The basic idea is to define the numerical algorithm using certain primitive flows, obtained from the original problem, which have the property that they can be integrated numerically to arbitrarily high order. It turns out that this provides sufficient structure to solve our problem and at the same time is general enough to find application to a variety of interesting examples as described below.

One approach to dealing with the constraint manifold is simply to introduce local coordinates and integrate the corresponding set of differential equations. This leads to difficulties, however, especially if trajectories of the system leave the coordinate chart where the local representation of the system is valid. To overcome this difficulty it has become common practice to embed the system, represented by a vector field on the manifold, into a higher-dimensional Euclidean space in which trajectories of the system can be represented globally. However, this presents a new problem in which the trajectories of an ordinary differential equation evolving in some Euclidean space are naturally constrained to some submanifold. A common approach here is simply to project back to the submanifold after one or several integration steps. Another approach is to treat the dynamical equations and the constraint equations as a differential-algebraic system. It would be useful, in these cases, to develop numerical integration procedures in which iterates of the integration algorithm always evolve on this submanifold. This is the approach we will take in this paper.

In the remainder of this introduction, we give a more technical description of our approach and give some examples. In Section 2 we recall some classical integration algorithms, and in Section 3 we introduce our corresponding versions of these algorithms. We shall only describe the rudimentary aspects of these algorithms but do not envisage any great difficulty in extending our work to the more refined algorithms met in practice. The main concept that we wish to develop in this paper is that of the order of an algorithm, since obtaining an algorithm of a particular order dictates the choice of many constants in the algorithm. To do this in the general setting that we have introduced here, we must study the approximation of flows of vector fields by the flows of a fixed frame $E_1, \ldots, E_n$; this we do in Sections 4 and 5. The general situation is considered in Section 4, while some particular computations with parameter-dependent vector fields are given in Section 5. In Section 6, we shall give specific details of third-order algorithms and relate them to the classical third-order algorithms. We make some concluding remarks in Section 7.

1.2. Vector Fields with Frozen Coefficients

Throughout this paper, we consider differential equations in the form

$$\dot{z} = F(t, z) = \sum_{i=1}^{n} f_i(t, z) E_i(z), \qquad z \in \mathbf{R}^N, \tag{1}$$

where $E_1, \ldots, E_n$ are real analytic vector fields on $\mathbf{R}^N$ and f_i are real analytic functions on $\mathbf{R} \times \mathbf{R}^N$. The basic idea we use is to define a numerical integration algorithm using flows of the "more elementary" system obtained by freezing the coefficients $f_i(t, z)$ to obtain a system of the form

$$\dot{z} = \sum_{i=1}^{n} a_i E_i(z), \qquad z \in \mathbf{R}^N. \tag{2}$$

Let $z(t) = \Lambda(t, a, z_0)$ denote the solution of this equation where defined, and satisfying $\Lambda(0, a, z_0) = z_0$, where $a = (a_1, \ldots, a_n) \in \mathbf{R}^n$. The principal assumption of this paper can be expressed in the following way:

Assumption 1. *The numerical values $\Lambda(t, a, z_0)$ may be computed "off line," to arbitrary accuracy for all $t \in \mathbf{R}$, $a \in \mathbf{R}^N$, $z_0 \in \mathbf{R}^N$.*

Formally, we define a vector field $F_{(\tau, p)}$ with *coefficients frozen at* (τ, p), relative to the frame $E_1, \ldots, E_n$ by

$$F_{(\tau, p)}(z) = \sum_{i=1}^{n} f_i(\tau, p) E_i(z), \qquad z \in \mathbf{R}^N. \tag{3}$$

Clearly Assumption 1 means that we may compute the solutions of the system (1), with "frozen" coefficients, to arbitrary accuracy. Our aim in this paper is to devise algorithms that compute solutions of the system (1) in terms of the solutions of the same system with "frozen" coefficients.

To demonstrate the link between this problem and the one stated earlier, concerning systems of ordinary differential equations evolving on manifolds, we introduce the Lie algebra L generated by the frame of vector fields $E_1, \ldots, E_n$. Let $L(z)$ denote the subspace of $\mathbf{R}^N$ spanned by all elements of L, evaluated at $z \in \mathbf{R}^N$. The mapping $z \mapsto L(z)$ is said to be a (real analytic) distribution on $\mathbf{R}^N$. A theorem of Herman/Nagano [21] shows that there exists a partition of $\mathbf{R}^N = \cup_\alpha M_\alpha$ by real analytic submanifolds of $\mathbf{R}^N$ such that for each $z \in M_\alpha$, $L(z)$ spans the tangent space to M_α at z. It follows that if $z_0 \in M_\alpha$, then

$$\Lambda(t, a, z_0) \in M_\alpha$$

for all $(t, a) \in (\mathbf{R} \times \mathbf{R}^N)$ where $\Lambda(t, a, z_0)$ is defined. Indeed, if the system (1) is initialized at $z_0 \in M_\alpha$, then its solution also evolves on M_α, and the vector field F may be restricted to the manifold M_α. It follows that any algorithms that compute solutions of the system (1) in terms of the system with "frozen" coefficients will naturally yield iterates that, when initialized at $z_0 \in M_\alpha$, evolve on M_α, at least to the accuracy with which the values of Λ may be computed.

As described above there is a considerable literature devoted to the numerical solution of differential-algebraic equations. This literature certainly includes discussion of numerical techniques dedicated to the solution of differential equations evolving on manifolds, described as the level sets of a set of given functions. This work, however, takes as its premise that the basic numerical integration methods remain the classical integration schemes and endeavors to specialize these schemes to the given situation. On the other hand, with respect to differential-algebraic systems, we are endeavoring to tackle the problem using an entirely new premise, where "simple" integration schemes that obey the constraints are already given, and the task becomes that of using these schemes to integrate more complicated systems, which also satisfy the constraints. Clearly, once this new class of algorithm has been established in works such as this, the algorithms need to be analyzed and compared to existing schemes, such as those for solving differential-algebraic systems. No attempt has been made to accomplish this second task in this paper, but it will be the focus of further papers by the authors.

It is important to note, however, that our algorithms can also be used when the simpler systems, defined by the vector fields with frozen coefficients, are not tangent to any manifold, and hence techniques for solving differential-algebraic systems are not applicable.

1.3. Two Examples

We shall give two example applications of equations with the structure described in equation (1). The first example may be written in the form

$$\dot{R} = S(f(R))R, \qquad R \in \mathcal{GL}(3, \mathbf{R}) \subset \mathbf{R}^{3\times 3}, \tag{4}$$

where $f(R) = (f_1(R), f_2(R), f_3(R))^T$, f_i are real-valued functions on $\mathcal{GL}(3, \mathbf{R})$, the group of nonsingular 3×3 matrices, and

$$S(a) = \begin{bmatrix} 0 & a_3 & -a_2 \\ -a_3 & 0 & a_1 \\ a_2 & -a_1 & 0 \end{bmatrix}$$

for every $a = (a_1, a_2, a_3)^T \in \mathbf{R}^3$. The skew symmetry of S implies that each solution of (4) satisfies the following constraints:

$$R^T(t)R(t) = R_0^T R_0, \qquad R(0) = R_0. \tag{5}$$

Defining the vector fields

$$\begin{aligned} E_1(R) &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} R, \qquad E_2(R) = \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} R, \\ E_3(R) &= \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} R \end{aligned} \tag{6}$$

we may rewrite equation (4) in the form

$$\dot{R} = \sum_{i=1}^{3} f_i(R) E_i(R). \tag{7}$$

Explicit formulas are available for the expressions $\Lambda(t, a, R_0)$ in this case [23]. Note that in the case $R_0 = I$, the 3×3 identity matrix, the solution of equation (4) evolves in the special orthogonal group $SO(3, \mathbf{R})$. The equations (4) may then represent the evolution of the attitude of a rigid body, although the function f is typically related to the angular velocity of the body [7]. The constraints (5) may be eliminated from the equation (4) to obtain a system of differential equations evolving in a local (three-dimensional) coordinate chart for $SO(3, \mathbf{R})$, typically taken to be defined by the Euler angles, [7]. However, singularities in these local coordinate versions of the equations force the use of equations such as those given in equation (4). Our aim is to use the formulas for Λ, obtained from the corresponding equations with "frozen" coefficients, relative to the frame E_1, E_2, and E_3 in (6), to integrate the equations (7).

With the next example, we wish to illustrate another application of our proposed algorithms. Instead of viewing the vector fields with frozen coefficients as tangent to a particular manifold, it is also useful to view them as just objects that are simpler to integrate than the original vector field. In the classical case the frozen coefficient vector fields are constant, and so integrating them is very easy. However, there are other vector fields that are relatively easy to integrate, such as linear vector fields. Certain linear vector fields may be integrated explicitly, as in the example above, while for others the flow may be approximated to arbitrary accuracy, with little effort.

The example is a version of Duffing's equation, considered in [8], whose exposition we follow here. We are not advocating integrating Duffing's equation with this

algorithm, but rather we include this example as a particularly simple application of our viewpoint. The version of Duffing's equation considered,

$$\begin{aligned} \dot{x}_1(t) &= x_2, \\ \dot{x}_2(t) &= -\eta_1 x_1 - \eta_2 x_1^3, \end{aligned} \tag{8}$$

is a simplified model describing the buckling of a beam, with a cubic nonlinearity. Here x_1 denotes the deflection of the beam from equilibrium, x_2 is the velocity of the beam at this point, η_1 and η_2 are positive constants.

We split the dynamics of the Duffing system into two pieces $F_0 + F_1$, where

$$\begin{aligned} F_0 &= x_2 \frac{\partial}{\partial x_1} - \eta_1 x_1 \frac{\partial}{\partial x_2}, \\ F_1 &= x_1 a(x) \frac{\partial}{\partial x_2}, \qquad a(x) = -\eta_2 x_1^2. \end{aligned}$$

Note that the system $\dot{x} = F_0(x(t))$ is a linear system and can be explicitly integrated. "Freezing" the coefficient $a(x)$ of F_1, yields a linear system F that is explicitly integrable involving simply the evaluation of a sine and cosine. Thus the expressions for Λ can easily be computed.

2. Classical Numerical Integration Algorithms

In this section we shall review some classical numerical integration algorithms for a system of differential equations written in the form

$$\dot{z} = F(z), z \in \mathbf{R}^N, \quad z(0) = z_0. \tag{9}$$

Note that we have not considered time-dependent vector fields F, as we did in equation (1), in order to simplify the notation. However, extension to time-dependent systems is straightforward. We first consider the Runge-Kutta methods described by the following equations.

2.1. Classical (Explicit) Runge-Kutta Algorithms

$$\begin{aligned} \nu_1(z) &= z, \\ \nu_2(z) &= z + hc_{2,1}F(z), \\ \nu_3(z) &= z + h(c_{3,1}F(z) + c_{3,2}F(\nu_2(z))), \\ &\vdots \\ \nu_r(z) &= z + h(c_{r,1}F(z) + c_{r,2}F(\nu_2(z)) + \cdots + c_{r,r-1}F(\nu_{r-1}(z))), \end{aligned} \tag{10}$$

$$\begin{aligned} z_{k+1} &= u_{k+1}(z_k, h) \\ &= z_k + h(c_1F(z_k) + c_2F(\nu_2(z_k)) + \ldots + c_rF(\nu_r(z_k))). \end{aligned} \tag{11}$$

The algorithm described by the equations (10) and (11) will be said to be an r-stage algorithm. We may also consider implicit r-stage algorithms by including in the expression for $v_k(z)$, terms like $c_{k,k+j}F(v_{k+j}(z)), 0 \leq j \leq r-k$.

Let us denote the flow of the vector field F by the mapping

$$(t, z) \longmapsto e^{tF}z, \qquad (t, z) \in R \times R^N,$$

where the exponential notation should not be given any explicit sense at present. Thus, the solution of equation (9) may be written as

$$z(t) = e^{tF}z_0. \tag{12}$$

We shall assume that the flow of F is defined for all $(t, z) \in \mathbf{R} \times \mathbf{R}^{N+1}$, for simplicity. For any particular choice of the constants $c_{k,j}, c_m, 2 \leq k \leq r, 1 \leq j < k, 1 \leq m \leq r$, which in the future we denote simply by c, we may define the order of the algorithm as follows:

The algorithm (10) and (11) has order q if for all $z \in \mathbf{R}^N$ and for all vector fields F, the Taylor series expansions of $u_{k+1}(z, h)$ and $e^{hF}z$ about $h = 0$ coincide up to and including terms involving h^q.

The equations obtained by equating powers of h may themselves be separated into equations involving independent differential operators formed from F, since these equations must hold for all F. These operators are sometimes called "elementary differentials" in the numerical analysis literature; see, for example, [19], page 145. These equations in turn yield equations for the constants c, which must hold in order for the algorithm to have a particular order.

In the case of explicit third-order algorithms, with three stages, $q = r = 3$, we obtain the following independent equations:

$$\begin{aligned}
&\text{(i)} \quad c_1 + c_2 + c_3 = 1, \\
&\text{(ii)} \quad c_2c_{21} + c_3(c_{31} + c_{32}) = 1/2, \\
&\text{(iii)} \quad \frac{c_2}{2}c_{21}^2 + \frac{c_3}{2}(c_{31} + c_{32})^2 = 1/6, \\
&\text{(iv)} \quad c_3c_{32}c_{21} = 1/6.
\end{aligned} \tag{13}$$

These equations have many solutions, as described in detail in [19]; see page 141. We have taken as a test case the following solution of the equations (13), attributed to "Kutta."

$$c_1 = \frac{1}{6}, \quad c_2 = \frac{2}{3}, \quad c_3 = \frac{1}{6}, \quad c_{31} = -1, \quad c_{32} = 2, \quad c_{21} = \frac{1}{6}. \tag{14}$$

The classical "Runge-Kutta" algorithms are fourth order, $q = 4$, and have four stages, $r = 4$. We now consider one class of multistep method, although other classes could be treated using the methods described in the paper with more effort.

2.2. *Classical (Explicit) Multistep Algorithms*

$$\begin{aligned} z_{k+1} &= u_{k+1}(z_k, z_{k-1}, \ldots, z_{k-r}) \\ &= z_k + h(\alpha_0 F(z_k) + \alpha_1 F(z_{k-1}) + \ldots + \alpha_r F(z_{k-r})). \end{aligned} \tag{15}$$

We say that the algorithm (15) has $r + 1$ steps. Implicit algorithms may be obtained by including terms

$$h\alpha_r F(z_{k-r}), \qquad r < 0$$

in the expression (15). In these cases we must rearrange the update equations in some way in order to solve for the most recent iterate. Again, for any particular choice of the constants $\alpha_k, 0 \leq k \leq r$, which in future we refer to as the constants α, we may define the order of the algorithm as follows:

The algorithm (15) has order q, if for all $z \in \mathbf{R}^N$ and for all vector fields F, the Taylor series expansions of

$$u_{k+1}(z, e^{-hF}z, \ldots, e^{-rhF}z)$$

and

$$e^{hF}z$$

about $h = 0$, coincide, up to and including terms involving h^q.

The same remarks, concerning the equations for the constants c, apply to the equations that determine the constants α.

In the case of explicit third-order algorithms, with 3 steps, $r = 2$, and order 3, $q = 3$, we obtain the following independent equations:

$$\begin{aligned} \alpha_0 + \alpha_1 + \alpha_2 &= 1, \\ -\alpha_1 - 2\alpha_2 &= \frac{1}{2}, \\ \alpha_1 + 4\alpha_2 &= \frac{1}{3}. \end{aligned} \tag{16}$$

These equations have a unique solution given by [19]:

$$\alpha_2 = \frac{5}{12}, \quad \alpha_1 = -\frac{4}{3}, \quad \alpha_0 = \frac{23}{12}. \tag{17}$$

3. Numerical Integration Algorithms Adapted to Frames

In this section we shall define our generalizations of the (explicit) Runge-Kutta and multistep algorithms described in the previous section. The adaptation to implicit algorithms is self-evident, even if this implementation would be more problematic than their classical counterparts. In this section we shall deal with an ordinary differential equation described by the equations

$$\dot{z} = F(z) = \sum_{i=1}^{n} f_i(z) E_i(z), \qquad z \in \mathbf{R}^N \tag{18}$$

with respect to a fixed frame $E_1, E_2 \ldots, E_n$ of vectors on $\mathbf{R}^N$. In particular, F_p will denote the vector field F with coefficients frozen at p, as described in Section 1. Note that, for simplicity, we will not treat time-varying vector fields F; this will not, however, make the following analysis conceptually more difficult. Our generalization of the Runge-Kutta algorithms may be described as follows.

3.1 Explicit Runge-Kutta Algorithms Adapted to a Frame

$$\begin{aligned}
\nu_1(p) &= p, \qquad F_p^1(h)(z) = F_p(z), \\
\nu_2(h, p) &= e^{hc_{2,1}F_p^1} p, \qquad F_p^2(h)(z) = F_{\nu_2(h,p)}(z), \\
\nu_3(h, p) &= e^{hc_{3,2}F_p^2} e^{hc_{3,1}F_p^1} p, \qquad F_p^3(h)(z) = F_{\nu_3(h,p)}(z), \\
&\vdots \\
\nu_r(h, p) &= e^{hc_{r,r-1}F_p^{r-1}} e^{hc_{r,r-2}F_p^{r-2}} \ldots e^{hc_{r,1}F_p^1} p, \qquad F_p^r(h)(z) = F_{\nu_r(h,p)}(z),
\end{aligned} \tag{19}$$

$$\begin{aligned}
u_{k+1}(h, p) &= e^{hc_r F_p^r} e^{hc_{r,1}F_p^{r-1}} \ldots e^{hc_1 F_p^1} p, \\
z_{k+1} &= u_{k+1}(h, z_k)
\end{aligned} \tag{20}$$

Note that $\nu_k(0, p) = p$, so that $F_p^k(0) = F_p$, and the update rule (20) is defined by flows of the vector field F with frozen coefficients. It follows that if the vector fields $E_1, \ldots, E_n$ are tangent to a submanifold M_α, as described in the introduction, then the iterates z_k will also evolve on M_α, assuming that $z_0 \in M_\alpha$.

Another important fact about the algorithm (19) and (20) is that it reduces to the classical algorithm under appropriate circumstances, namely,

$$E_k(z) = \frac{\partial}{\partial z_k}, \qquad 1 \le k \le n = N. \tag{21}$$

We shall denote the conditions (21) as the Euclidean case. Let e_k denote the kth standard basis vector in $\mathbf{R}^N$. Then under the conditions (21) we have

$$e^{tE_k} p = p + te_k,$$

and identifying vector fields on $\mathbf{R}^N$ as vectors in $\mathbf{R}^N$ as usual, we may write

$$F_{\nu_k(h,p)} = F_p^k(h) = F(\nu_k(h, p)).$$

It follows that in the Euclidean case (21), the algorithm (19) and (20) does indeed reduce to the classical algorithm (10) and (11). We now describe our multistep algorithm.

3.2. *Explicit Multistep Algorithm Adapted to a Frame*

$$u_k^j(h, z_k, z_{k-1}, \ldots, z_{k-r}, u_k^{j+1}) = e^{h\alpha_0^j F_{z_k}} e^{h\alpha_1^j F_{z_{k-1}}} \ldots e^{h\alpha_r^j F_{z_{k-r}}} u_k^{j+1},$$
$$(0 \leq j \leq l-1),$$
$$u_k^l = z_k, \tag{22}$$
$$z_{k+1} = u_k^0(h, z_k, z_{k-1}, \ldots, z_{k-r} u_k^1).$$

In this paper we shall be mostly concerned with the case $l = 2$, although the general case may be analyzed using similar techniques. In the Euclidean case (21) we easily establish that

$$u_k^j\left(h, z_k, z_{k-1}, \ldots, z_{k-r}, u_k^{j+1}\right)$$
$$= h\left(\alpha_0^j F(z_k) + \alpha_1^j F(z_{k-1}) + \cdots + \alpha_r^j F(z_{k-r})\right) + u_k^{j+1}.$$

It follows that under the identification

$$\alpha_i = \sum_{k=0}^{l-1} \alpha_i^k, \qquad 0 \leq i \leq r,$$

the algorithm (22) does reduce to that of the classical algorithm with update rule given by equation (15).

Our next step is to identify a unified setting in which to examine the order of both of the algorithms identified in this section. This may be achieved by defining vector fields G_k, $1 \leq k \leq s$, as follows:

$$G_k(h, p)(z) = \sum_{j=1}^{n} g_j^k(h, p) E_j(z), \tag{23}$$

where g_j^k are analytic functions on $\mathbf{R} \times \mathbf{R}^N$. Construct a mapping $\Psi : \mathbf{R} \times \mathbf{R}^N \to R^N$ as follows

$$\Psi(h, p) = e^{G_1(h,p)} e^{G_2(h,p)} \ldots e^{G_s(h,p)} p. \tag{24}$$

Assume that the coefficients g_j^k are determined in some fixed way from only the coefficients f_i and vector fields E_k, as well as $(h, p) \in \mathbf{R} \times \mathbf{R}^N$; note that F, defined in equation (18), depends only on f_i and E_k. We are interested in simultaneously approximating all vector fields F while the frame of vector fields $E_1, \ldots, E_n$ is fixed. We therefore make the following definition:

The mapping Ψ simultaneously approximates all of the vector fields F (relative to the frame $E_1, \ldots, E_n$), to order q, if for all $p \in \mathbf{R}^N$ and all choices of coefficient functions f_i the Taylor series expansions of $\Psi(h, p)$ and $e^{hF} p$ about $h = 0$ coincide up to and including terms including h^q.

Using this definition, we may define the notion of order for the algorithms introduced above, simply by identifying the approximate mappings Ψ. In the case of the Runge-Kutta algorithms (19) and (20), we simply set

$$\Psi(h, p) = u_{k+1}(h, p), \tag{25}$$

in which case the vector fields G_k can be identified with $hc_jF_p^i(h)$ for some $j, 1 \leq j \leq r$. The algorithm (19) and (20) has order q, if and only if the mapping Ψ in (25) simultaneously approximates all of the vector fields F to order q.

In the case of the multistep algorithms (22) the mapping Ψ is defined by setting

$$\Psi(h, p) = u_k^0\left(h, p, e^{-hF}p, \ldots, e^{-rhF}p, u_k^1\right), \tag{26}$$

in which case the vector fields G_k can be identified with $h\alpha_i^j F_{e^{-ihF}}$ for some $j, 0 \leq j \leq l-1$ and some $i, 0 \leq i \leq r$. The algorithm (22) has order q, if and only if the mapping Ψ in (26) simultaneously approximates all of the vector fields F to order q.

It follows that we may investigate the problem of choosing the constants c or α to obtain algorithms of order q, by first analyzing the problem of simultaneous approximation of vector fields F by general mappings Ψ in (24) and then specializing to the two cases described above. Sections 4 and 5 of this paper deal with the general problem and Section 6 deals with the specific cases where $q = 3$.

We finally remark that the algorithms identified above have been described in $\mathbf{R}^N$. However, if the vector fields $E_1, \ldots, E_n$ are tangent to a manifold M_α, as described in the introduction, our invariant description of the algorithm enables us to view them as iteration processes on M_α itself. The definition of the order of algorithm can now be given, just as above, by noting that we may view Ψ in (24) as a mapping

$$\Psi : \mathbf{R} \times M_\alpha \to M_\alpha.$$

In this case, however, the notion of Taylor series, as used above, is no longer well defined. This notion is modified by choosing any coordinate mapping ψ

$$\psi : M_\alpha \to \mathbf{R}^{\dim(M_\alpha)}$$

and considering the mappings $\psi \circ \Psi : R \times M_\alpha \to \mathbf{R}^{\dim(M_\alpha)}$, whose Taylor series in h about $h = 0$ are well defined. The definition of simultaneous approximation now becomes the following:

The mapping Ψ simultaneously approximates all of the vector fields F (relative to the frame $E_1, \ldots, E_n$ on the manifold M_α) to order q if, for all $p \in M_\alpha$, and a coordinate mapping ψ with domain including p, and all choices of coefficient functions $F_i|_{M_\alpha}$, the Taylor series expansions of $\psi \circ \Psi(h, p)$ and $\psi(e^{hF}p)$ about $h = 0$ coincide up to and including terms including h^q.

4. Simultaneous Approximation of Flows by Flows of Fixed Frame

In this section we study the problem of determining when a mapping Ψ, as in equation (24), approximates all of the vector fields F, relative to a fixed frame $E = \{E_1, \ldots, E_n\}$, to order q. In particular, we are interested in computing the Taylor series expansions about $h = 0$ of the expressions $e^{hF}p$ and $\Psi(h, p)$.

Thus, we must find necessary and sufficient conditions for the following relations to hold:

$$\frac{d^k}{dh^k}\psi(e^{hF}p)\bigg|_{h=0} = \frac{d^k}{dh^k}\psi \circ \Psi(h, p)\bigg|_{h=0}, \qquad 1 \leq k \leq q, \tag{27}$$

over all coefficient functions $(f_1, \ldots, f_n)$, $p \in \mathbf{R}^N$. We let $F(\psi)$ denote the Lie derivative of ψ by F, and then set

$$F^k(\psi) = F^{k-1}(F(\psi)), \qquad F^1(\psi) = F(\psi).$$

It follows that the Taylor series expansions of $\psi(e^{hf}\, p)$ about $h = 0$ may be written in the form

$$\psi(e^{hF}\, p) = \psi(p) + hF(\psi)(p) + \frac{h^2}{2!}F^2(\psi)(p) + \cdots. \tag{28}$$

In order to express the differential operators F^k in a convenient form and calculate the derivatives on the right-hand side of equation (27), we must introduce some further notation. Let X_k denote one of the vector fields $E_1, \ldots, E_n$, so that $X_k = E_{i_k}$ for $1 \leq i_k \leq n$. Let X^α denote the differential operator defined by the equation

$$X^\alpha(\psi) = X_{i_1}X_{i_2} \ldots (X_{i_k}(\psi)) \ldots),$$

where $\alpha = (i_1, \ldots, i_k)$ is a multi-index of length $|\alpha| = k$. We say D is a differential operator based on the frame E if

$$D(\psi)(z) = \sum_\alpha d_\alpha(z)X^\alpha(\psi)(z),$$

where d_α is an analytic function on $\mathbf{R}^N$, for each multi-index α. ($d_\alpha \equiv 0$ for all but a finite number of indices α.) If

$$D(\psi) = \sum_{\substack{|\alpha| = k \\ \alpha}} d_\alpha X^\alpha(\psi),$$

then we say that D has order k (relative to E). We introduce two operations on the set of differential operators based on the frame E. Let D_1 and D_2 be two differential operators based on the frame E with orders k_1 and k_2. Then

$$(D_1 \cdot D_2)(\psi)(z) = \sum_{\substack{|\alpha_1| = k_1 \\ |\alpha_2| = k_2}} d^1_{\alpha_1}(z)\, d^2_{\alpha_2}(z)X^{\alpha_1}(X^{\alpha_2}(\psi))(z),$$

and

$$(D_1(D_2))(\psi)(z) = \sum_{\substack{|\alpha_1| = k_1 \\ |\alpha_2| = k_2}} d^1_{\alpha_1}(z)X^{\alpha_1}(d^2_{\alpha_2})(z)X^{\alpha_2}(\psi)(z).$$

Thus $D_1 \cdot D_2$ has order $k_1 + k_2$ and $D_1(D_2)$ has order k_2. Recalling that

$$F(\psi) = \sum_{i=1}^n f_i E_i(\psi),$$

we see that F is a first-order differential operator based on the frame E. By Leibnitz rule we have

$$\begin{aligned} F^2(\psi) &= (F(F))(\varphi) + (F \cdot F)(\psi), \\ F^3(\psi) &= (F(F(F)))(\psi) + 2(F \cdot F(F))(\psi) \\ &\quad + (F(F) \cdot F)(\psi) + (F \cdot F \cdot F)(\psi), \end{aligned}$$

and so on.

We introduce the notation

$$F^{(k)} = \underbrace{F(F(F(\cdots(F)\cdots)}_{k}$$

to denote the first-order component of F^k, relative to E. Thus,

$$\begin{aligned} F^2 &= F^{(2)} + F \cdot F, \\ F^3 &= F^{(3)} + 2F \cdot F^{(2)} + F^{(2)} \cdot F + F \cdot F \cdot F, \end{aligned} \tag{29}$$

and so on.

In general, if D is a differential operator, $D = \sum_\alpha d_\alpha X^\alpha$, then we set

$$(D_p\psi)(z) = \sum_\alpha d_\alpha(p) X^\alpha(\psi)(z).$$

D_p is the operator D with coefficients "frozen" at p.

A differential operator is said to be of depth l if it is in the linear span of the following set of operators D:

D is a composition of l vector fields, under the operations '$\cdot$' and '$(\cdot)$,' chosen from the set $\{f_1E_1, \ldots, f_nE_n\}$ where $\{f_1, \ldots, f_n\}$ is a fixed set of real analytic coefficient functions.

Let $V(F)$ denote the vector space of all differential operators generated by the single vector field $F = \sum_{i=1}^n f_iE_i$, under the operations '$\cdot$' and '$(\cdot)$'. We denote the subspace of $V(F)$ composed of differential operators of order k and depth l by $V_k^l(F)$. Note that this space plays the role of the vector space of elementary differentials in the Euclidean case; see [19]. We similarly denote the vector spaces of differential operators, obtained by freezing the coefficients of operators in $V(F)$ and $V_k^l(F)$ at p, by $V(F)(p)$ and $V_k^l(F)(p)$. We set

$$V^l(F) = V_1^l(F) + V_2^l(F) + \ldots + V_l^l(F)$$

and hence, also define $V^l(F)(p)$, by freezing coefficients at p. We also set

$$\begin{aligned} L^1(F)(p) &= V_1^1(F)(p), \\ L^l(F)(p) &= V_1^l(F)(p) + \sum_{k=1}^{l-1} [V_l^k(F)(p), L^{l-k}(F)(p)]. \end{aligned} \tag{30}$$

We have $F^l \in V^l(F), F^{(l)} \in V_1^l(F)$, and $F_p^l \in V^l(F)(p), F_p^{(l)} \in V_1^l(F)(p)$, as well as the identities

$$V_r^l(F) \cdot V_s^k(F) \subset V_{r+s}^{k+1}(F), \qquad V_r^l(F)(p) \cdot V_s^k(F)(p) \subset V_{r+s}^{k+1}(F)(p),$$

$$V_r^l(F)(V_s^k(F)) \subset V_s^{l+k}(F),$$

$$L^l(F)(p) \subset V^l(F)(p).$$

If $F = \sum_{i=1}^n f_i E_i$, we set $\hat{F} = (f_1, \ldots, f_n) \in C^\omega(R^N)^n$ and denote by $D(\hat{F})$ any element of $V(F)$.

We denote by V the vector space of all mappings D defined by the assignments:

$$\hat{F} \longmapsto D(\hat{F}), \tag{31}$$

under pointwise addition. Similarly, V_k^l and V^l will denote those subspaces of V obtained by restricting $D(\hat{F})$ in (31) to lie in $V_k^l(F)$ and $V^l(F)$, respectively. There is a natural notion of linear independence in V : if $D^1 \ldots, D^\tau$, are elements of V, then $D^1, \ldots, D^\tau$ are linearly independent if for all $r-$tuples of real numbers $(\gamma_1, \ldots, \gamma_r)$

$$\begin{aligned} &\sum_{i=1}^{\tau} \gamma_i D^i(\hat{F})(\psi) \equiv 0, \qquad \forall \psi \in C^\omega(\mathbf{R}^N), \quad \hat{F} \in C^\omega(\mathbf{R}^N)^n \\ &\Longrightarrow \gamma_1 = \gamma_2 = \cdots = \gamma_r = 0. \end{aligned} \tag{32}$$

We denote by $V(p)$ the vector space of mappings

$$\hat{F} \mapsto D_p(\hat{F}),$$

under pointwise addition, where D_p is any operator in $V(F)(p)$. We obtain subspaces of $V(p)$ by restricting the range of the mapping above, for each fixed p, to $V^l(F)(p)$, $V_k^l(F)(p)$ and $L^l(F)(p)$, and denote them by $V^l(p)$, $V_k^l(p)$, and $L^l(p)$, respectively. We extend the notion of linear independence in these spaces: if $D_p^1, \ldots, D_p^r$ are elements of $V(p)$, then $D_p^1, \ldots, D_p^r$ are linearly independent, if for all r-tuples of real numbers $(\gamma_1, \ldots, \gamma_r)$

$$\begin{aligned} &\sum_{i=1}^{r} \gamma_i D_p^i(\hat{F})(\psi)(p) \equiv 0, \qquad \forall \psi \in C^\omega(\mathbf{R}^N), \quad p \in \mathbf{R}^N, \quad \hat{F} \in C^\omega(\mathbf{R}^N)^n \\ &\Longrightarrow \gamma_1 = \gamma_2 = \cdots = \gamma_r = 0. \end{aligned} \tag{33}$$

It is interesting to list spanning sets for the spaces V^l and $L^l(p)$ for small l. By slight abuse of notation we shall use F to denote the mappings in V defined by

$$\hat{F} \longmapsto \sum_{i=1}^{n} f_i E_i.$$

We have the following spanning sets for V_k^l

$$V_1^1 = \mathrm{Sp}\{F\}, \quad V_1^2 = \mathrm{Sp}\{F(F)\}, \quad V_2^2 = \mathrm{Sp}\{F \cdot F\},$$

$$
\begin{aligned}
V_1^3 &= \mathrm{Sp}\{(F(F))(F), (F \cdot F)(F)\} \qquad V_2^3 = \mathrm{Sp}\{F \cdot F(F), F(F) \cdot F\},\\
V_3^3 &= \mathrm{Sp}\{F \cdot F \cdot F\},\\
V_1^4 &= \mathrm{Sp}\{((F(F)(F))(F), ((F \cdot F)(F))(F), (F \cdot F(F))(F), (F(F) \cdot F)(F)\\
&\quad (F \cdot F \cdot F)(F)\},\\
V_2^4 &= \mathrm{Sp}\{((F(F))(F)) \cdot F, ((F \cdot F)(F)) \cdot F, (F(F)) \cdot (F(F)), F \cdot ((F \cdot F)(F))\\
&\quad F \cdot (F(F)(F))\},\\
V_3^4 &= \mathrm{Sp}\{F(F) \cdot F \cdot F, F \cdot F(F) \cdot F, F \cdot F \cdot F(F)\},\\
V_4^4 &= \mathrm{Sp}\{F \cdot F \cdot F \cdot F\},
\end{aligned}
\tag{34}
$$

and so on.

It is interesting to observe that the operators $F^{(l)} \in V_1^l(F)$ may be easily expressed in terms of these spanning sets, as follows:

$$
\begin{aligned}
F^{(1)} &= F,\\
F^{(2)} &= F(F),\\
F^{(3)} &= F(F(F)) = (F \cdot F)(F) + (F(F))(F),\\
F^{(4)} &= (F \cdot F \cdot F)(F) + 2(F \cdot F(F))(F) + (F(F) \cdot F)(F)\\
&\quad + ((F(F))(F))(F) + ((F \cdot F)(F))(F).
\end{aligned}
\tag{35}
$$

Spanning sets for V^l, $V_k^l(p)$, and $V^l(p)$ may be constructed in a similar way. In particular, from equation (30) and (34) we may write down spanning sets for $L^l(p)$ for small l as follows:

$$
\begin{aligned}
L^1(p) &= \mathrm{Sp}\{F_p\} = V_1^1(p),\\
L^2(p) &= \mathrm{Sp}\{F(F)_p\} = V_1^2(p),\\
L^3(p) &= \mathrm{Sp}\{V_1^3(p), [F_p, F(F)_p]\},\\
L^4(p) &= \mathrm{Sp}\{V_1^4(p), [F_p, V_1^3(p)], [F_p, [F_p, F(F)_p]]\},
\end{aligned}
\tag{36}
$$

and so on.

The spanning sets of V^l, $V^l(p)$, and $L^l(p)$ as described above are not necessarily independent in the sense of definitions (32) or (33), depending on the structure of the fixed frame $E = \{E_1, \ldots, E_n\}$. However, for the purposes of this paper we make the following assumption:

Assumption 2. *The spanning sets of V^l, $V^l(p)$, and $L^l(p)$ generated as in (34) and (36) from F are maximally independent in the sense of definitions (32) and (33) respectively.*

We note that in the Euclidean case (21), all of the Lie brackets occurring in the spanning sets for $L^l(p)$ vanish, since they involve only Lie brackets of the frame $E = \{\partial/\partial_{z_1}, \ldots, \partial/\partial_{z_N}\}$. Thus in the Euclidean case we have

$$
L^l(p) = V_1^l(p). \tag{37}
$$

In general we set $n^l = \dim V^l = \dim V^l(p), N^l = \dim L^l(p)$ and $n_k^l = \dim V_k^l = \dim V_k^l(p)$. Thus, we have $n^l = \sum_{k=1}^{l} n_k^1$ and $n_1^l = N^l$ if condition (37) holds. We now define a sequence of differential operators:

$$\begin{aligned} \Gamma_{k+1} &= F(\Gamma_k) + F \cdot \Gamma_k + F \cdot F^{(k)}, \qquad k \geq 1, \\ \Gamma_1 &= 0. \end{aligned} \tag{38}$$

Thus,

$$\Gamma_2 = F \cdot F, \qquad \Gamma_3 = F(F \cdot F) + F \cdot F \cdot F + F \cdot F^{(2)} = F^{(2)} \cdot F + 2F \cdot F^{(2)} + F \cdot F \cdot F$$

and so on. We may deduce that

$$\Gamma_k = \Gamma_k(F, F^{(2)}, \ldots, F^{(k-1)}),$$

and because of Γ_k's structure we have

$$\Gamma_k(F, F^{(2)}, \ldots, F^{(k-1)})(\psi)(p) = \Gamma_k(F_p, F_p^{(2)}, \ldots, F_p^{(k-1)})(\psi)(p). \tag{39}$$

By comparing equations (28), (29), (30), and (39) we deduce the following result:

Lemma 1.

$$\left.\frac{d^k}{dh^k}\psi(e^{hF}p)\right|_{h=0} = [F_p^{(k)} + \Gamma_k(F_p, F_p^{(2)}, \ldots, F_p^{(k-1)})](\psi)(p). \tag{40}$$

□

We now define a parameter-dependent vector field $A(h)$ by setting

$$\frac{d}{dh}\psi \circ \Psi(h, p) = (A(h)\psi) \circ \Psi(h, p). \tag{41}$$

We note that each derivative $\partial^k A(h)/\partial h^k$ is also a vector field, which we denote by $A(h)^{(k)}$, with

$$A(h) = A(h), \quad A(h)^{(1)} = A(h), \quad A(h)^{(2)} = \ddot{A}(h), \quad A(h)^{(3)} = \dddot{A}(h).$$

As we will show in Section 6, for the classes of algorithms introduced in the previous section, we have

$$A(0)^{(k)} \in L^{k+1}(F)(p), \qquad k \geq 0. \tag{42}$$

Viewed as a differential operator based on the frame $E, A(h)^{(k)}$ has frozen coefficients. We therefore calculate

$$\frac{d^2}{dh^2}\psi \circ \Psi(h, p) = (A(h) \cdot A(h))(\psi) \circ \Psi(h, p) + \dot{A}(h)(\psi) \circ \Psi(h, p),$$

$$\begin{aligned} \frac{d^3}{dh^3}\psi \circ \Psi(h, p) = {} & (A(h) \cdot A(h) \cdot A(h))(\psi) \circ \Psi(h, p) + \ddot{A}(h)(\psi) \circ \Psi(h, p) \\ & + 2(A(h) \cdot \dot{A}(h))(\psi) \circ \Psi(h, p) + (\dot{A}(h) \cdot A(h))(\psi) \circ \Psi(h, p) \end{aligned}$$

From equations (28) and (29) and Lemma 1, the calculations above demonstrate the following result:

Lemma 2.

$$\frac{d^k}{dh^k}\psi \circ \Psi(h, p)\Big|_{h=0} = [A(0)^{(k-1)} + \Gamma_k(A(0), A(0)^{(1)}, \ldots, A(0)^{(k-2)})](\psi)(p). \quad (43)$$

□

From the definition of the spaces $V^k(F)(p)$ and $L^k(F)(p)$, it is clear that Lemmas 1 and 2 express their respective derivatives as expressions $(D\psi)(p)$ where $D \in V^k(F)(p)$. Recalling that equation (27) requires that these expressions are equated over all $\hat{F} \in C^\omega(\mathbf{R}^N)^n$, $\psi \in C^\omega(\mathbf{R}^N)$ and points $p \in \mathbf{R}^N$, our definition of independence of differential operators in $V^k(p)$ shows that the sufficient conditions (27) for a qth-order algorithm are equivalent to equating the coefficients of independent differential operators in the right-hand sides of equations (40) and (43); that is:

$$A(0)^{(k-1)} + \Gamma_k(A(0), A(0)^{(1)}, \ldots, A(0)^{(k-2)}) = F_p^{(k)} + \Gamma_k(F_p, F_p^{(2)}, \ldots, F_p^{(k-1)}), \quad 1 \le k \le q, \quad (44)$$

viewed as expressions in $V^{(k)}(p)$. Our assumption (42) and the independence assumption (Assumption # 2), together with equation (30), imply that we may decompose $A(0)^{(k-1)}$ uniquely into two components

$$A(0)^{(k-1)} = A_p^k + B_p^k, \quad (45)$$

where

$$A_p^k \in V_1^k(p)$$

and

$$B_p^k \in \sum_{j=1}^{k-1} [V_1^j(p), L^{k-j}(p)] \subset V_2^k(p) \oplus V_3^k(p) \oplus \cdots \oplus V_k^k(p).$$

Further, from the definitions of Γ_k we observe that

$\Gamma_k(A(0), A(0)^{(1)}, \ldots, A(0)^{(k-2)})$ and $\Gamma_k(F_p, F_p^{(2)}, \ldots, F_p^{(k-1)})$ both lie in the space $V_2^k(p) \oplus V_3^k(p) \oplus \cdots V_k^k(p)$. We therefore, by equating the differential operators in (44) for $1 \le k \le q$ and invoking the independence assumptions, obtain the following result.

Theorem 1. *Under the Assumption 2, necessary and sufficient conditions for a mapping Ψ to approximate all of the vector fields F, relative to a fixed frame $E = \{E_1, \ldots, E_n\}$ to order q, are given by the following two equivalent conditions:*

(i) (a) $A_p^k = F_p^{(k)}$ *as elements in* $V_1^k(p)$, $\quad 1 \le k \le q.$ (46)

(b) $B_p^k = 0$ *as elements in* $\sum_{j=1}^{k-1} [V_1^j(p), L^{k-j}(p)]$, $\quad 2 \le k \le q.$ (47)

$$\text{(ii) (a) } A_p^k = F_p^{(k)} \text{ as elements in } V_1^k(p), \qquad 1 \le k \le q.$$

$$\text{(b) } B_p^k + \Gamma_k(A_p^1 + B_p^1, A_p^2 + B_p^2, \ldots, A_p^{k-1} + B_p^{k-1}) = \Gamma_k(F_p, F_p^{(2)}, \ldots, F_p^{(k-1)}), \qquad 2 \le k \le q \tag{48}$$

$$\text{as elements in } V_1^k(p) \oplus V_2^k(p) \oplus \cdots \oplus V_k^k(p). \qquad \square$$

Under our independence assumption (Assumption #2), the conditions (i)(a) yields n_1^k equations, while (i)(b) yields $N^k - n_1^k$ equations; and the condition (ii)(a) again yields n_1^k equations, while (ii)(b) yields $\sum_{j=2}^k n_j^k$ equations. Inspection of the spanning sets in equations (34) and (36) demonstrate that in general

$$N^k - n_1^k \ll \sum_{j=2}^{k} n_j^k \quad \text{or} \quad N^k \ll n^k.$$

The total number of equations obtained by using the conditions (i) is $\sum_{k=1}^q N^k$, while using the conditions (ii), the total number of equations is $\sum_{k=1}^q n^k$.

Note that in the Euclidean case (21) conditions (i)(b) and (ii)(b) are vacuous, since for condition (i), equation (37) holds, and for condition (ii) $B_p^k = 0$ implies that the resulting condition (ii)(b) is satisfied as a consequence of condition (ii)(a). Thus, in the Euclidean case we simply have the conditions

$$A_p^k = F_p^{(k)}, \qquad 1 \le k \le q. \tag{49}$$

These can be simply interpreted as the classical conditions, as we demonstrate in Section 6.

In the next section we deal with the problem of obtaining expressions for the operators $A(0)^{(k)}$, and A_p^k, B_p^k, in terms of the vector fields G_k, and their derivatives.

5. Calculus of Parameter-Dependent Vector Fields

The definition of the differential operator $A(h)$ is given in terms of the mapping Ψ, as described in equation (24), as follows:

$$\frac{d}{dh}\psi \circ \Psi(h, p) = (A(h)\psi) \circ \Psi(h, p).$$

The purpose of this section is to characterize $A(h)$ and its derivatives in terms of the operators $G_k(h, p)$ that define Ψ. We have the following expansion:

$$\psi(e^{tG_k(h,p)}p) = \sum_{i=0}^{\infty} (G_k(h, p)^i\psi)(p)(t^i/i!),$$

with the series converging for small $|t|$ and appropriate conditions on the coefficient functions of $G_k(h, p) = \sum_{j=1}^n g_j^k(h, p)E_j$. Note that $G_k(0, p)$ is the zero vector field. Rather than use the geometric interpretation of $(p, t) \longmapsto e^{tG_k(h,p)}p$ as the flow

of the vector field G_k, it is more convenient to use its interpretation as a formal sum of differential operators. Because of the reduced importance of the parameter p, in this section we shall suppress it in many expressions; thus we set

$$z_t^k(h) = \sum_{i=0}^{\infty} (t^i/i!) G_k(h)^i.$$

Thus, in suitable domains (and conditions on the coefficient functions of G_k) we have the following:

$$z_{-t}^k(h)(\psi)(p) = \psi(e^{tG_k(h)} p). \tag{50}$$

If G and H are arbitrary vector fields, we set

$$\begin{aligned} \operatorname{ad} G(H) &= [G, H] = \operatorname{ad}^1 G(H), \\ \operatorname{ad}^i G(H) &= \operatorname{ad} G(\operatorname{ad}^{i-1} G(H)), \qquad i \geq 2, \\ \operatorname{Ad} z_t^k(h)(G) &= \sum_{i=0}^{\infty} (t^i/i!) \operatorname{ad}^i G_k(h)(G). \end{aligned}$$

We may view $\operatorname{Ad} z_t^k(h)$ as a formal isomorphism of the Lie algebra of analytic vector fields, its series expansion converging on suitable domains. The following result is responsible for this latter observation, and again holds on suitable domains of convergence:

$$\operatorname{Ad} z_t^k(h)(G)(\psi)(p) = d\psi(e^{-tG_k(h)} * G(e^{tG_k(h)} p)),$$

where $*$ is the differential operator of mappings. We set (as we did for the operator $A(h)$)

$$\frac{\partial_i}{\partial h^i} G_k(h) = G_k(h)^{(i)}, \qquad G_k(h)^{(0)} = G_k(h)$$

and

$$G_k(h)^{(1)} = \dot{G}_k(h) \quad G_k(h)^{(2)} = \ddot{G}_k(h) \quad G_k(h)^{(3)} = \dddot{G}_k(h).$$

We may now define another formal differential operator by setting

$$(G_t^k(h, p) =) G_t^k(h) = \int_0^t \operatorname{Ad} z_s^k(h)(\dot{G}_k(h))\, ds. \tag{51}$$

Denote the derivatives of G_t^k with respect to h, in the same way as we did for the derivatives of G_k and A. We now obtain two intermediate results.

Lemma 3. *As formal differential operators, $\psi \in C^\omega(\mathbf{R}^N)$, $p \in \mathbf{R}^N$*

$$\frac{\partial}{\partial h} z_t^k(h)(\psi)(p) = \operatorname{Ad} z_{-t}^k(h)(G_t^k(h))(\psi)(e^{tG_k(h)} p), \tag{52}$$

or analytically

$$\frac{\partial}{\partial h} e^{tG_k(h)} p = \int_0^t e^{(t-s)G_k(h)} * \dot{G}_k(h)(e^{sG_k(h)} p)\, ds. \tag{53}$$

Proof.

$$\frac{\partial}{\partial t} \psi(e^{tG_k(h)} p) = G_k(h)(\psi)(e^{tG_k(h)} p)$$

by definition of the flow of a vector field. It follows that

$$\frac{\partial}{\partial t} d\psi \left(\frac{\partial}{\partial h} e^{tG_k(h)} p \right) = \dot{G}_k(h)(\psi)(e^{tG_k(h)} p) + d(G_k(h)(\psi)) \left(\frac{\partial}{\partial h} e^{tG_k(h)} p \right).$$

It follows that $(\partial/\partial h e^{tG_k(h)} p)$ satisfies a differential equation, whose solution is seen to be given by equation (53), by uniqueness of solutions of differential equations. Equation (52) is simply a restatement of equation (53), in terms of the formal differential operators z_t^k and G_t^k. □

Lemma 4. *If G is an arbitrary vector field, $\psi \in C^\omega(\mathbf{R}^N)$, $p \in \mathbf{R}^N$, as formal differential operators*

$$\frac{\partial}{\partial h} \mathrm{Ad}\, z_{-t}^k(h)(G)(\psi) = -\mathrm{Ad}\, z_{-t}^k(h)(\mathrm{ad}\, G_t^k(h)(G))(\psi), \tag{54}$$

or analytically

$$\frac{\partial}{\partial h} e^{tG_k(h)} * G(e^{-tG_k(h)} p) = -e^{tG_k(h)} * \int_0^t [e^{-sG_k(h)} * \dot{G}_k(h) \circ e^{sG_k(h)}, G](e^{-tG_k(h)} p)\, ds. \tag{55}$$

Proof. Let $z(h, t, p)(\psi) = e^{tG_k(h)} * G(e^{-tG_k(h)} p)(\psi)$. Thus,

$$\frac{\partial z}{\partial t}(h, t, p)(\psi) = z(h, t, p)(G_k(h)(\psi)) - G_k(h)(p)(z(h, t, \cdot)(\psi)).$$

Hence,

$$\begin{aligned} \frac{\partial}{\partial t} \frac{\partial z}{\partial h}(h, t, p)(\psi) &= \frac{\partial z}{\partial h}(h, t, p)(G_k(h)(\psi)) - G_k(h)(p) \left(\frac{\partial z}{\partial h}(h, t, .)(\psi) \right) \\ &\quad + z(h, t, p)(\dot{G}_k(h)(\psi)) - \dot{G}_k(h)(p)(z(h, t, \cdot)(\psi)) \\ &= -\mathrm{ad}\, G_k(h) \left(\frac{\partial z}{\partial h}(h, t, \cdot) \right)(\psi)(p) - \mathrm{ad}\, \dot{G}_k(h)(z(h, t, \cdot))(\psi)(p). \end{aligned}$$

By uniqueness of solutions of differential equations, we find that

$$\frac{\partial z}{\partial h}(h, t, p)(\psi) = -\int_0^t e^{G_k(h)(t-s)} * [\dot{G}_k(h), z(h, s, \cdot)](e^{-G_k(h)(t-s)} p)(\psi)\, ds.$$

Using the fact that $\operatorname{Ad} z_s^k(h)$ is a Lie algebra isomorphism, the above expression reduces to equation (55). The expression (54) is simply a restatement of equation (55) using the formal differential operators z_t^k and G_t^k. □

We are now in a position to compute the derivative in equation (41). Note that we may express the mapping Ψ in equation (24) in terms of the operators z_t^k as follows:

$$\psi \circ \Psi(h, p) = z^s(h)(z^{s-1}(h)(\ldots(z^1(h)(\psi)\ldots)(p)$$

where

$$z^k(h) = z_1^k(h), \qquad 1 \leq k \leq s.$$

Similarly we set

$$G^k(h) = G_1^k(h), \qquad 1 \leq k \leq s$$

and

$$\pi^k(h) = \operatorname{Ad} z_{-1}^1(h) \circ \operatorname{Ad} z_{-1}^2(h) \circ \cdots \circ \operatorname{Ad} z_{-1}^k(h)$$

and

$$A^k(h) = \pi^k(h)(G^k(h)). \tag{56}$$

We may now describe the derivative in equation (41) in terms of these operators.

Lemma 5.

$$\frac{\partial}{\partial h}\psi \circ \Psi(h, p) = \sum_{k=1}^{s} A^k(h)(\psi)(\Psi(h, p)). \tag{57}$$

Proof. By Lemma 3,

$$\begin{aligned}
&\frac{\partial}{\partial h_k} z^k(h_k)(z^{k-1}(h)(\ldots(z^1(h)(\psi)\ldots)(e^{G_{k+1}(h)}e^{G_{k+2}(h)}\ldots e^{G_s(h)}p)\Big|_{h_k=h} \\
&\qquad = \operatorname{Ad} z_{-1}^k(h)(G^k(h))(z^{k-1}(h)(\ldots z^1(h)(\psi)\ldots)(e^{G_k(h)}\ldots e^{G_s(h)}p) \\
&\qquad = \operatorname{Ad} z_{-1}^1(h)(\operatorname{Ad} z_{-1}^2(h)(\ldots \operatorname{Ad} z_{-1}^{k-1}(h)(\operatorname{Ad} z_{-1}^k(h_k)(G^k(h)\ldots)(\psi) \\
&\qquad\quad \circ (\Psi(h, p)).
\end{aligned}$$

Now

$$\begin{aligned}
\frac{\partial}{\partial h}\psi \circ \Psi(h, p) &= \sum_{k=1}^{s} \frac{\partial}{\partial h_k} z^k(h_k)(\ldots z^1(h)(\psi)\ldots)(e^{G_{k+1}(h)}\ldots e^{G_s(h)}p)\Big|_{h_k=h} \\
&= \sum_{k=1}^{s} \pi^k(h)(G^k(h)(G^k(h))(\psi)(\Psi(h, p)).
\end{aligned}$$

The derivatives of the operators $A^k(h)$ with respect to h are denoted as for the operators $A(h)$, $G_k(h)$, and $G_{(t)}^k(h)$. We have a preliminary result in this regard:

Lemma 6. *For any vector field G*

$$\frac{\partial}{\partial h}\pi^k(h)(G) = -\sum_{j=1}^{k}[\pi^j(h)(G^j(h)), \pi^k(h)(G)]. \tag{58}$$

Proof.

$$\frac{\partial}{\partial h}\pi^k(h)(G) = \sum_{j=1}^{k}\frac{\partial}{\partial h_j}\mathrm{Ad}\, z_{-1}^1(h)(\ldots \mathrm{Ad}\, z_{-1}^j(h_j)(\ldots \mathrm{Ad}\, z_{-1}^k(h)(G)\ldots))$$

$$(\text{evaluated at } h_j = h).$$

By Lemma 4, this may be evaluated as

$$-\sum_{j=1}^{k}\pi^{j-1}(h)(\mathrm{Ad}\, z_{-1}^j(h)(\mathrm{ad}\, G^j(h)\{\mathrm{Ad}\, z_{-1}^{j+1}(h)(\ldots \mathrm{Ad}\, z_{-1}^k(h)(G)\ldots)\})).$$

Since $\mathrm{Ad}\, z_t^j(h)$ are Lie algebra isomorphisms, we may write this expression in the form

$$-\sum_{j=1}^{k}\pi^{j-1}(h)(\mathrm{ad}\,(\mathrm{Ad}\, z_{-1}^j(h)(G^j(h))\{\mathrm{Ad}\, z_{-1}^j(h)(\ldots \mathrm{Ad}\, z_{-1}^k(h)(G)\ldots)\})).$$

Iterating this observation yields the required expression (58). □

Corollary 1.

$$\frac{\partial}{\partial h}A^k(h) = -\sum_{j=1}^{k-1}[\pi^j(h)(G^j(h)), \pi^k(h)(G^k(h))] + \pi^k(h)(\dot{G}^k(h)). \tag{59}$$

Proof. Apply Lemma 6 to the case where $G = G^k(h)$, and note that the final term in the series vanishes since the Lie bracket of two equal vector fields is zero. □

We may now apply these results to obtain a description of the derivatives $A^k(h)^{(l)}$.

Lemma 7. *$A^k(h)^{(l)}$ is a Lie polynomial in the quantities $\pi^k(h)G^m(h)^{(j)}$, $1 \le m \le k$, $1 \le j \le l$. In particular, $A^k(h)^{(l+1)}$ is obtained from $A^k(h)^{(l)}$ using the substitution:*

$$\pi^k(h)(G^m(h)^{(j)}) \mapsto \pi^k(h)G^m(h)^{(j+1)} - \sum_{i=1}^{m}[\pi^i(h)(G^i(h)), \pi^k(h)(G^m(h)^{(j)})]. \tag{60}$$

□

Our main goal is to derive formulas for the derivatives $A^k(0)^{(l)}$ and hence $A(0)^{(l)}$. Since we assume $G_k(h, p) = G_k(0) = 0$, the operators $z_t^k(h)$, $\mathrm{Ad}\, z_t^k(h)$ and $\pi^k(h)$, all reduce to the identity operators at $h = 0$. Thus we may derive from Lemma 7 the following special cases.

Corollary 2. *$A^k(0)^{(l)}$ is a Lie polynomial in the quantities $G^m(0)^{(j)}$, $1 \le m \le k$, $1 \le j \le l$. In particular, $A^k(0)^{(l+1)}$ is obtained from $A^k(0)^{(l)}$ using the substitution:*

$$G^m(0)^{(j)} \mapsto G^m(0)^{(j+1)} - \sum_{i=1}^{m} [G^i(0), G^m(0)^{(j)}]. \tag{61}$$

□

Corollary 3.

$$\dot{A}^k(0) = \dot{G}^k(0) - \sum_{j=1}^{k-1} [G^j(0), G^k(0)], \tag{62}$$

$$\begin{aligned} \ddot{A}^k(0) = {} & \ddot{G}^k(0) + [\dot{G}^k(0), G^k(0)] + \sum_{j=1}^{k-1} (2[\dot{G}^k(0), G^j(0)] + [G^k(0), \dot{G}^j(0)]) \\ & + \sum_{j=1}^{k-1} [G^j(0), [G^j(0), G^k(0)]] + 2 \sum_{m=2}^{k-1} \sum_{j=1}^{m-1} ([G^j(0), [G^m(0), G^k(0)]]). \end{aligned} \tag{63}$$

Proof. Formula (62) follows directly from (59) by evaluating at $h = 0$. Formula (63) can be derived by using the substitution (61) in the formula (62). This gives

$$\begin{aligned} \ddot{A}^k(0) = {} & \ddot{G}^k(0) - \sum_{j=1}^{k-1} [\dot{G}^j(0), G^k(0)] - \sum_{j=1}^{k-1} [G^j(0), \dot{G}^k(0)] - \sum_{j=1}^{k} [G^j(0), \dot{G}^k(0)] \\ & + \sum_{j=1}^{k-1} \sum_{m=1}^{j-1} [[G^m(0), G^j(0)], G^k(0)] + \sum_{j=1}^{k-1} \sum_{m=1}^{k-1} [G^j(0), [G^m(0), G^k(0)]]. \end{aligned}$$

This expression reduces to that in equation (63) after simplifying using the Jacobi identity. □

Lemma 7 and Corollary 2 reduce the problem of evaluating $A^k(0)^{(l)}$ to one of evaluating the derivatives $G^k(0)^{(l)}$, or $G_1^k(0)^{(l)}$, where $G_t^k(h)$ is defined by formula (51). Evaluating these derivatives turns out to be fairly complicated, so we only compute the first two derivatives, without obtaining explicit formulas for the general case. From (51) and (54) we obtain

$$\begin{aligned} \dot{G}^k(h) = {} & -\int_0^1 \operatorname{Ad} z_s^k(h)[G_{-s}^k(h), \dot{G}_k(h)]\, ds + \int_0^1 \operatorname{Ad} z_s^k(h) \ddot{G}_k(h)\, ds \\ = {} & \int_0^1 \int_0^s \operatorname{Ad} z_s^h(h)[\operatorname{Ad} z_{-\sigma}^k(h)(\dot{G}_k(h)), \dot{G}_k(h)]\, d\sigma\, ds \\ & + \int_0^1 \operatorname{Ad} z_s^k(h)(\ddot{G}_k(h))\, ds. \end{aligned}$$

Differentiating with respect to h once more gives

$$\begin{aligned}\ddot{G}^k_{(h)} = &\int_0^1 \mathrm{Ad}\, z^k_s(h)(\dddot{G}_h(h))\, ds - \int_0^1 \mathrm{Ad}\, z^k_s(h)([G^k_{-s}(h), \ddot{G}_k(h)])\, ds \\ &+ \int_0^1 \int_0^s \{\mathrm{Ad}\, z^k_s(h)([\mathrm{Ad} z^k_{-\sigma}(h)(\ddot{G}_k(h)), \dot{G}_k(h)]) \\ &+ \mathrm{Ad}\, z^k_s(h)([\mathrm{Ad}\, z^h_{-\sigma}(h)(\dot{G}_k(h)), \ddot{G}_k(h)]) \\ &- \mathrm{Ad}\, z^k_s(h)([G^k_{-s}(h), [\mathrm{Ad}\, z^k_{-\sigma}(h)(\dot{G}_k(h))]]) \\ &- \mathrm{Ad}\, z^k_s(h)([\mathrm{Ad}\, z^k_{-\sigma}(h)([G^k_\sigma(h), \dot{G}_k(h)]), \dot{G}_k(h)])\}\, d\sigma\, ds.\end{aligned}$$

Thus,

$$\begin{aligned}\ddot{G}^k_{(h)} = &\int_0^1 \mathrm{Ad}\, z^k_s(h)(\dddot{G}_k(h))\, ds \\ &+ \int_0^1 \int_0^s \{\mathrm{Ad}\, z^k_s(h)([\mathrm{Ad}\, z^k_{-\sigma}(h)(\ddot{G}_k(h)), \dot{G}_k(h)]) \\ &+ 2\mathrm{Ad}\, z^k_s(h)([\mathrm{Ad}\, z^k_{-\sigma}(h)(\dot{G}_k(h)), \ddot{G}_k(h)])\}\, d\sigma\, ds \\ &+ \int_0^1 \int_0^s \int_0^\sigma \{\mathrm{Ad}\, z^k_s(h)([\mathrm{Ad}\, z^k_{-\sigma}(h)(\dot{G}_k(h)), [\mathrm{Ad}\, z^k_{-\tau}(h)(\dot{G}_k(h)), \dot{G}_k(h)]]) \\ &+ \mathrm{Ad}\, z^k_s(h)([\mathrm{Ad}\, z^k_{-\tau}(h)(\dot{G}_k(h)), [\mathrm{Ad}\, z^k_{-\sigma}(h)(\dot{G}_k(h)), \dot{G}_k(h)]]) \\ &+ \mathrm{Ad}\, z^k_s(h)([\dot{G}_k(h), \mathrm{Ad}\, z^k_{-\sigma}(h)([\mathrm{Ad}\, z^k_\tau(h)(\dot{G}_k(h)), \dot{G}_k(h)])])\}\, \delta\tau\, d\sigma\, ds.\end{aligned} \tag{64}$$

From these expressions we deduce the following result.

Lemma 8. *$G^k(0)^{(l)}$ is a Lie polynomial in $G_k(0)^{(j)}$, $1 \le j \le l+1$, with no constant term, with linear term equal to $G_k(0)^{(l+1)}$ and higher-order terms involving only $G_k(0)^{(j)}$, $1 \le j \le l$. In particular, we have*

$$\begin{aligned}G^k(0) &= \dot{G}_k(0), \qquad \dot{G}^k(0) = \ddot{G}_k(0), \\ \ddot{G}^k(0) &= \dddot{G}_k(0) + \frac{1}{2}[\dot{G}_k(0), \ddot{G}_k(0)].\end{aligned} \tag{65}$$

□

We may now put together the results of Corollary 3 and Lemma 8 to give explicit formulas for the lowest derivatives of $A^k(h)$.

Corollary 4.

$$\begin{aligned}A^k(0) &= \dot{G}_k(0), \\ \dot{A}^k(0) &= \ddot{G}_k(0) - \sum_{j=1}^{k-1} [\dot{G}_j(0), \dot{G}_k(0)],\end{aligned}$$

$$\begin{aligned}\ddot{A}^k(0) = \dddot{G}_k(0) &+ \frac{1}{2}[\ddot{G}_k(0), \dot{G}_k(0)] \\ &+ \sum_{j=1}^{k-1}([\dot{G}_k(0), \ddot{G}_j(0)] + 2[\ddot{G}_k(0), \dot{G}_j(0)]) \\ &+ \sum_{j=1}^{k-1}[\dot{G}_j(0), [\dot{G}_j(0), \dot{G}_k(0)]] \\ &+ \sum_{m=2}^{k-1}\sum_{j=1}^{m-1} 2([\dot{G}_j(0), [\dot{G}_m(0), \dot{G}_k(0)]]).\end{aligned} \tag{66}$$

6. Specific Examples of Algorithms on Manifolds

In this section we shall apply the analysis of the previous two sections to obtain third-order Runge-Kutta and multistep algorithms, adapted to a frame E, as explained in Section 3. The analysis will also demonstrate that third-order algorithms are the first instance of algorithms where the non-Euclidean structure of the frame E, plays a nontrivial role; or in other words, all second-order Euclidean algorithms are second-order with respect to any frame.

To apply the analysis of Sections 5 and 4 we need only compute the derivatives $G_k(0)^{(l)}$ using the specific form of the algorithm in question. We first consider the Runge-Kutta algorithms. For third-order algorithms we consider three-stage algorithms as described by equations (19) and (20), explicitly:

$$\begin{aligned}\nu_1(p) &= p, \qquad F_p^1(h)(z) = F_p(z), \\ \nu_2(h, p) &= e^{hc_{2,1}F_p^1}p, \qquad F_p^2(h)(z) = F_{\nu_2(h,p)}(z), \\ \nu_3(h, p) &= e^{hc_{3,2}F_p^2}e^{hc_{3,1}F_p^1}p, \qquad F_p^3(h)(z) = F_{\nu_3(h,p)}(z), \\ z_{k+1} &= e^{hc_3F_{z_k}^3}e^{hc_2F_{z_k}^2}e^{hc_1F_{z_k}^1}z_k = u_{k+1}(h, z_k).\end{aligned} \tag{67}$$

Thus, as in the definition of Ψ for the Runge-Kutta algorithms, equation (25), we obtain

$$G_1(h, p) = hc_3F_p^3(h), \quad G_2(h, p) = hc_2F_p^2(h), \quad G_3(h, p) = hc_1F_p^1(h). \tag{68}$$

Lemma 9. *For the Runge-Kutta algorithm (67) we have*

$$\begin{aligned}\dot{F}_p^1(0) &= 0, \qquad \ddot{F}_p^1(0) = 0, \\ \dot{F}_p^2(0) &= c_{2,1}F(F)_p, \qquad \dot{F}_p^3(0) = (c_{3,2} + c_{3,1})F(F)_p, \\ \ddot{F}_p^2(0) &= c_{2,1}^2(F \cdot F)(F)_p, \\ \ddot{F}_p^3(0) &= 2c_{3,2}c_{2,1}(F(F))(F)_p + (c_{3,1} + c_{3,2})^2(F \cdot F)(F)_p.\end{aligned}$$

Proof. We apply the general expansion (28), while noting that the vector fields F_p^k have "frozen coefficients," to obtain

$$F_p^2 = F_p^1 + hc_{2,1}F_p^1(F)_p + (hc_{2,1})^2\frac{1}{2!}(F_p^1 \cdot F_p^1)(F)_p + o(h^2),$$

$$\begin{aligned} F_p^3 = {} & F_p^1 + hc_{3,2}F_p^2(F)p + (hc_{3,2})^2\frac{1}{2!}(F_p^2 \cdot F_p^2)(F)_p \\ & + h_{3,1}F_p^1(F)_p + (hc_{3,1})^2\frac{1}{2!}(F_p^1 \cdot F_p^1)(F)_p \\ & + h^2c_{3,1}c_{3,2}(F_p^1 \cdot F_p^2)(F)_p + o(h^2). \end{aligned}$$

Since F_p^1 does not depend on h, we obtain

$$\begin{aligned} \dot{F}_p^2 = {} & c_{2,1}F_p^1(F)_p + hc_{2,1}^2(F_p^1 \cdot F_p^1)(F)_p + o(h), \\ \dot{F}_p^3 = {} & c_{3,2}F_p^2(F)_p + hc_{3,2}^2(F_p^2 \cdot F_p^2)(F)_p + c_{3,1}F_p^1(F)_p \\ & + hc_{3,1}^2(F_p^1 \cdot F_p^1)(F)_p + 2hc_{3,1}c_{3,2}(F_p^1 \cdot F_p^2)(F)_p \\ & + hc_{3,2}\dot{F}_p^2(F)_p + h^2c_{3,1}c_{3,2}(F_p^1 \cdot \dot{F}_p^2)(F)_p \\ & + (hc_{3,2})^2\frac{1}{2!}((\dot{F}_p^2 \cdot F_p^2)(F)_p + (F_p^2 \cdot \dot{F}_p^2)(F)_p) + o(h). \end{aligned}$$

Noting that $F_p^2(0) = F_p^1 = F_p$ we obtain

$$\begin{aligned} \dot{F}_p^2(0) &= c_{2,1}F(F)_p, \qquad \dot{F}_p^3(0) = (c_{2,1} + c_{3,1})F(F)_p, \\ \ddot{F}_p^2(0) &= (c_{2,1})^2(F \cdot F)(F)_p, \\ \ddot{F}_p^3(0) &= (c_{3,1} + c_{3,2})^2(F \cdot F)(F)_p + 2c_{3,2}\dot{F}_p^2(0)(F)_p. \end{aligned}$$

Finally, substituting the value of $\dot{F}_p^2(0)$, we obtain the stated result. □

Using the formulas for the vector fields $G_k(h, p)$ in equation (68) we deduce from Lemma 9 the following result.

Corollary 5. *For the Runge-Kutta algorithm (67) we have*

$$\begin{aligned} &\dot{G}_3(0, p) = c_1F_p, \quad \dot{G}_2(0, p) = c_2F_p, \quad \dot{G}_1(0, p) = c_3F_p, \\ &\ddot{G}_3(0, p) = \dddot{G}_3(0, p) = 0, \\ &\ddot{G}_2(0, p) = 2c_2c_{2,1}F(F)_p, \quad \dddot{G}_2(0, p) = 3c_2c_{2,1}^2(F \cdot F)(F)_p, \\ &\ddot{G}_1(0, p) = 2c_3(c_{3,2} + c_{3,1})F(F)_p, \\ &\dddot{G}_1(0, p) = 6c_3c_{32}c_{31}(F(F))(F)_p + 3c_3(c_{3,1} + c_{3,2})^2(F \cdot F)(F)_p. \end{aligned}$$ □

Note that we could obtain the expressions for the derivatives $F_p^k(0)^{(l)}$ by applying the theory of Sections 4 and 5 to the expressions $F_{\nu_k(h,p)}(\psi)$, rather than the expression

$\psi \circ \Psi(h, p)$. However, the preceding analysis demonstrates the type of computation that those sections are generalizing.

We may now use the result of Corollary 5 in Corollary 4 to obtain explicit formulas for the derivatives $A(0)^{(l)} = \sum_{k=1}^{s} A^k(0)^{(l)}$, in the case $s = 3$, $l = 0$, 1, and 2. Much simplification is obtained since $\dot{G}_k(0)$ are all multiples of the same vector field F_p.

Corollary 6. *For the Runge-Kutta algorithm (67), we have*

$$\begin{aligned} A(0) &= (c_1 + c_2 + c_3)F_p, \dot{A}(0) = 2(c_2c_{2,1} + c_3(c_{3,2} + c_{3,1}))F(F)_p, \\ \ddot{A}(0) &= 3(c_2c_{2,1}^2 + c_3(c_{3,1} + c_{3,2})^2)(F \cdot F)(F)_p \\ &\quad + 6c_3c_{3,2}c_{2,1}(F(F))(F)_p \\ &\quad + \{4c_2c_3c_{2,1} + c_2^2c_{2,1} + c_3^2(c_{3,2} + c_{3,1}) \\ &\quad - 2(c_1c_2c_{2,1} + c_3(c_1 + c_2)(c_{3,2} + c_{3,1}))\}[F(F)_p, F_p]. \end{aligned} \tag{69}$$

We now repeat these calculations for the multistep algorithm (22). For third-order algorithms we consider only the case $l = 2$, with 3 steps, that is:

$$z_{k+1} = e^{h\alpha_0^0 F_{z_k}} e^{h\alpha_1^0 F_{z_{k-1}}} e^{h\alpha_2^0 F_{z_{k-2}}} e^{h\alpha_0^1 F_{z_k}} e^{h\alpha_1^1 F_{z_{k-1}}} e^{h\alpha_2^1 F_{z_{k-2}}} z_k. \tag{70}$$

Thus, using the definition (26) of $\Psi(h, p)$ for multistep algorithms we have $s = 6$ and

$$\begin{aligned} G_1(h, p) &= h\alpha_0^0 F_p, G_2(h, p) = h\alpha_1^0 F_{e^{-hF}p}, \\ G_3(h, p) &= h\alpha_2^0 F_{e^{-2hF}p}, G_4(h, p) = h\alpha_0^1 F_p, \\ G_5(h, p) &= h\alpha_1^1 F_{e^{-hF}p}, G_6(h, p) = h\alpha_2^1 F_{e^{-2hF}p}. \end{aligned} \tag{71}$$

Lemma 10.

$$\frac{\partial^j}{\partial h^j} hF_{e^{-mHF}p}\bigg|_{h=0} = jF_p^{(j)}(-m)^{j-1}. \tag{72}$$

Proof. By induction and the definition of the vector field $F_p^{(j)}$ (Section 4) one can prove

$$\frac{\partial^j}{\partial h^j}(hF_{e^{-mhF}p}) = j(-m)^{j-1}F^{(j)}_{e^{-mhF}p} + h(-m)^j F^{(j+1)}_{e^{-mhF}p}, \qquad j \geq 1.$$

Equation (72) is obtained by evaluating at $h = 0$. □

The derivatives of $G_k(h, p)$ in equation (71), with respect to h, are now easily obtained from Lemma 10. We may now obtain the required derivatives $A(0)^{(l)} = \sum_{k=1}^{s} A^k(0)^{(l)}$ in the case of the multistep algorithm (70) by applying the results of Corollary 4, Lemma 10, and equations (71).

Lemma 11. *For the multistep algorithm (70) we have*

$$\begin{aligned}
A(0) &= (\alpha_0^0 + \alpha_1^0 + \alpha_2^0 + \alpha_0^1 + \alpha_1^1 + \alpha_2^1)F_p, \\
\dot{A}(0) &= -2(\alpha_1^0 + \alpha_1^1 + 2(\alpha_2^0 + \alpha_2^1))F_p^{(2)}, \\
\ddot{A}(0) &= 3(\alpha_1^0 + \alpha_1^1 + 4(\alpha_2^0 + \alpha_2^1))F_p^{(3)} \\
&\quad + \{((\alpha_1^0)^2 + (\alpha_1^1)^2 + 2((\alpha_2^0)^2 + (\alpha_2^1)^2)) \\
&\quad 2(\alpha_0^0(2\alpha_1^0 + 4\alpha_2^0 + 2\alpha_1^1 + 4\alpha_2^1) + \alpha_1^0(3\alpha_2^0 - \alpha_0^1 + \alpha_1^1 + 3\alpha_2^1) \\
&\quad + \alpha_2^0(-2\alpha_0^1 + 2\alpha_2^1) + \alpha_2^1) + \alpha_0^1(2\alpha_1^1 + 4\alpha_2^1) + \alpha_1^1(3\alpha_2^1))\}[F_p, F_p^{(2)}].
\end{aligned} \tag{73}$$

□

We wish to apply Theorem 1 to Corollary 6 and Lemma 11 in order to obtain algorithms adapted to an arbitrary frame E. However, this first requires that we verify the claim $A(0)^{(k)} \in L^{k+1}(F)(p); k \geq 0$. To do this, we introduce the notion of weight for parameter (h) dependent vector fields and their Lie polynomials. Referring to Corollary 2, we say that the vector fields $G^m(0)^{(j)}$ have weight $j+1$ and that a Lie monomial in these vector fields has weight equal to the sum of the weights of the constituent vector fields. A Lie element that is the sum of Lie monomials all of the same weight j is said to have weight j also.

Similarly, we can define weight for Lie monomials in the vector fields $G_m(0)^{(j)}$, by insisting that $G_m(0)^{(j)}$ has weight j. A Lie element that is a sum of Lie monomials in the vector fields $G_m(0)^{(j)}$, all of weight k, is said to have weight k also.

From Corollaries 2 and 3, we see that $A(0)^{(j)}$ is a Lie element of weight $(j+1)$, with respect to $G^m(0)$, while Lemma 8 and the preceding calculations demonstrate that $G^m(0)^{(j)}$ has weight j, with respect to $G_k(0)$. It follows (see Corollary 4), that $A(0)^{(j)}$ is a Lie element of weight $(j+1)$, with respect to the vector fields $G_k(0)$.

Now Lemma 10 and Corollary 5 show that for the Runge-Kutta and multistep algorithms adapted to a frame, we have that

$$G_k(0, p)^{(j)} \in V_1^j(F)(p).$$

$A(0)^{(j)} \in L^{j+1}(F)(p)$ now follows from the definitions of $L^j(F)(p)$ and the fact, as demonstrated above, that $A(0)^{(j)}$ has weight $(j+1)$, with respect to the vector field $G_k(0, p) = G_k(0)$.

To apply Theorem 1 we must obtain the decompositions

$$A(0)^{(k-1)} = A_p^k + B_p^k,$$

as in equation (45). Under the Assumption 2, and the fact that in both algorithms (67) and (70), $\dot{G}_k(0)$ are multiples of F_p, Corollary 4 gives

$$\begin{aligned}
A_p^1 &= \sum_{k=1}^{3} \dot{G}_k(0), A_p^2 = \sum_{k=1}^{3} \ddot{G}_k(0), A_p^3 = \sum_{k=1}^{3} \dddot{G}_p(0), \\
B_p^1 &= B_p^2 = 0, \\
B_p^3 &= \sum_{k=1}^{3} \left(\frac{1}{2}[\ddot{G}_k(0), \dot{G}_k(0)] + \sum_{j=1}^{k-1} ([\dot{G}_k(0), \ddot{G}_j(0)] + 2[\ddot{G}_k(0), \dot{G}_j(0)]) \right).
\end{aligned} \tag{74}$$

Theorem 2. *Under Assumption 2 the Runge-Kutta algorithm adapted to a frame E, given in equation (67), is of order 3 if and only if the Euclidean equations (13) are satisfied together with any one of the following equations:*

$$\text{(i)}\quad 4c_2c_3c_{2,1} + c_2^2c_{2,1} + c_3^2(c_{3,2} + c_{3,1}) - 2(c_1c_2c_{2,1} + c_3(c_1 + c_2)(c_{3,2} + c_{3,1})) = 0. \tag{75}$$

$$\text{(ii)}\quad 3(c_2^2c_{2,1} + c_3^2(c_{3,2} + c_{31})) + 6c_3(c_1 + c_2)(c_{3,2} + c_{3,1}) + 6c_1c_2c_{2,1} = 2.$$

$$\text{(iii)}\quad 3(c_2^2c_{2,1} + c_3^2(c_{3,2} + c_{3,1})) + 6c_2c_3c_{2,1} = 1.$$

The three sets of equations obtained by adjoining one of the equations (i), (ii), or (iii) to the Euclidean Runge-Kutta equations (13) have identical solutions.

Proof. Applying Theorem 1, part (i) to Corollary 5, and equations (74), or equivalently Corollary 6, yields the equations (13) and equation (75(i)).

Alternatively, we may apply part (ii) of Theorem 1. We obtain the equations (13) again by applying part (ii)(a) for $1 \leq k \leq 3$. Part (ii)(b) is vacuous for $k = 1, 2$. For $k = 3$ we have

$$B_p^3 + \Gamma_3(A_p^1, A_p^2) = \Gamma_3(F_p, F_p^{(2)}).$$

But from equations (38) we may rewrite this in the form $B_p^3 + A_p^2 \cdot A_p^1 + 2A_p^1 \cdot A_p^2 +$ $A_p^1 \cdot A_p^1 \cdot A_p^1 = F_p^{(2)} \cdot F_p + 2F_p \cdot F_p^{(2)} + F_p \cdot F_p \cdot F_p$, which reduces to the condition

$$B_p^3 + A_p^2 \cdot A_p^1 + 2A_p^1 \cdot A_p^2 = F_p^{(2)} \cdot F_p + 2F_p \cdot F_p^{(2)}. \tag{76}$$

Using Corollary 5 and equation (74) we may rewrite the left-hand side of equation (76) in the form

$$3(\dot{G}_2(0) \cdot \ddot{G}_1(0) + \dot{G}_3(0) \cdot \ddot{G}_1(0) + \dot{G}_3(0) \cdot \ddot{G}_2(0) + \ddot{G}_2(0) \cdot \dot{G}_1(0)) + \frac{3}{2}(\dot{G}_2(0) \cdot \ddot{G}_2(0) + \dot{G}_1(0) \cdot \ddot{G}_1(0) + \ddot{G}_2(0) \cdot \dot{G}_2(0) + \ddot{G}_1(0) \cdot \dot{G}_1(0)).$$

Using this result and Corollary 5 once more, equating coefficients of the (independent) vector fields $F_p \cdot F_p^{(2)}$ and $F_p^{(2)} \cdot F_p$ gives the equations (75)(ii) and (iii) respectively.

However, equations (75)(ii) and (iii) are not independent, given the equations (13); indeed we have

$$(75)(ii) + (75)(iii) = (13)(i) \times (13)(ii) \times 6.$$

Moreover, we have

$$3 \times (75)(i) = 2 \times (75)(iii) - (75)(ii).$$

This proves all assertions of Theorem 2. □

Applying "Mathematica" to the solution sets of equations in Theorem 2 yields five distinct solutions for the constants c; two one-parameter families depending on c_3 and three distinct isolated solutions. These are summarized in the table on page 30.

We now consider the multistep algorithm (70).

TABLE 1. Constants Defining Runge-Kutta Algorithms Adapted to Arbitrary Frames

	Classical "kutta" Algorithm	Runge-Kutta Algorithms Adapted to Arbitrary Frames				
		SOLN1	SOLN2	SOLN3	SOLN4	SOLN5
		$c_3 = 1$	$c_3 = 1$			
c_1	1/6	0.264	−1.264	1	13/51	7/24
c_2	2/3	−0.264	1.264	−2/3	24/17	−1/24
c_3	1/6	1	1	2/3	−2/3	3/4
$c_{2,1}$	1/2	0.858	−0.0583	−1/24	289/456	24/17
$c_{3,1}$	−1	0.532	3.434	161/24	21669/21964	1079/1836
$c_{3,2}$	2	0.194	−2.860	−6	−114/289	17/108

Theorem 3. *Under assumption (2) the multistep algorithm adapted to a frame E, given in equation (70), is of order 3 if and only if the following equations are satisfied:*

$$\text{(i)}\quad \alpha_0^0 + \alpha_1^0 + \alpha_2^0 + \alpha_1^1 + \alpha_2^1 = 1.$$

$$\text{(ii)}\quad \alpha_1^0 + \alpha_1^1 + 2(\alpha_2^0 + \alpha_2^1 = -\frac{1}{2}$$

$$\text{(iii)}\quad \alpha_1^0 + \alpha_1^1 + 4(\alpha_2^0 + \alpha_2^1) = \frac{1}{3}. \tag{77}$$

$$\begin{aligned}\text{(iv)}\quad &\frac{(\alpha_1^0)^2 + (\alpha_1^1)^2}{2} + (\alpha_2^0)^2 + (\alpha_2^1)^2 + \alpha_0^0(2\alpha_1^0 + 4\alpha_2^0 + 2\alpha_1^1 + 4\alpha_2^1)\\ &+ \alpha_1^0 + (3\alpha_2^0 - \alpha_0^1 + \alpha_1^1 + 3\alpha_2^1) + \alpha_2^0(-2\alpha_0^1 + 2\alpha_2^1)\\ &+ \alpha_0^1(2\alpha_1^1 + 4\alpha_2^1) + \alpha_1^1(3\alpha_2^1) = 0.\end{aligned}$$

Proof. We apply Theorem 1, part (i) to Lemma 11 and equations (74) directly. The equations (77) are obtained as required. □

One notes that if we set

$$\beta_0 = \alpha_0^0 + \alpha_0^1 \quad \beta_1 = \alpha_1^0 + \alpha_1^1 \quad \beta_2 = \alpha_2^0 + \alpha_2^1,$$

then if the constants α satisfy equations (77)(i), (ii), and (iii), the constants β satisfy the equations (16) for the Euclidean, third-order, multistep algorithms. The equations (77) therefore do indeed generalize the equations (16) for the Euclidean algorithms.

Applying "Mathematica" to the solution of the equations (77) yields three distinct sets of solutions, two of which are complex. The real solution is parameterized by α_1^1 and α_2^1 as follows:

$$\alpha_0^0 = \frac{17 + 216\alpha_1^1 + 552\alpha_2^1 + 144\alpha_1^1\alpha_2^1}{72(1 + 2\alpha_1^1 + 4\alpha_2^1)},$$

$$\alpha_1^0 = \frac{-4 - 3\alpha_1^1}{3}, \qquad \alpha_2^0 = \frac{5 - 12\alpha_2^1}{12},$$

$$\alpha_0^1 = \frac{121 + 60\alpha_1^1 - 144\alpha_1^1\alpha_2^1}{72(1 + 2\alpha_1^1 + 4\alpha_2^1)}.$$

In particular, we have a solution

$$\alpha_0^0 = \frac{17}{72}, \quad \alpha_1^0 = -\frac{4}{3}, \quad \alpha_2^0 = \frac{5}{12}, \quad \alpha_0^1 = \frac{121}{72}, \quad \alpha_1^1 = \alpha_2^1 = 0.$$

Note that

$$\alpha_0^0 + \alpha_0^1 = \frac{23}{12}, \quad \alpha_1^0 = -\frac{4}{3}, \quad \alpha_2^0 = \frac{5}{12},$$

which corresponds to the unique solution of the Euclidean third-order multistep equations (16) as given in (17).

We note the Theorems 2 and 3 illustrate two distinct means of achieving algorithms adapted to arbitrary frames.

7. Concluding Remarks

Clearly the work above represents only the beginning of a vast program of work, replicating the usual analysis of numerical integration algorithms for ODEs, Butcher [2,3], and others [19.20]. However as is clear from Section 6, any attempt to analyze algorithms of order greater than three will be very complex, and our current work is aimed at precisely this problem. We make some observations concerning this issue:

7.1. Computing the Constraint Equations

The derivation of the equations constraining the c constants in the Runge-Kutta algorithms, cf. Theorem 2, and the α constants in the case of the multistep algorithms, cf. Theorem 3, involve implicit computations of the higher-order derivations F^k, $k \geq 0$, as well as certain operators with "frozen" coefficients. There are several ways in which these computations can be organized. One means is to keep careful track of the spanning sets V_k^l. Another method is to use the calculus of rooted trees, labeled with the vector field F as in [9, 17]. It turns out that the vector space, whose basis is the set of such trees, has an algebraic structure that can be exploited to yield efficient algorithms for manipulating the higher-order derivations F^k. Some of the computations in this paper were done in this way using trees. This point of view is exploited systematically in the companion paper [10]. We have implemented the code to manipulate trees corresponding to the higher-order derivations in Maple, Mathematica, and Snobol.

7.2. Solving the Constraint Equations

Solving the constraint equations for the c or α constants has been done so far using an ad hoc mixture of symbolic and numeric techniques. At present, we know how to do this only for low-order systems and are currently developing algorithms to solve these equations for higher-orders.

7.3. Integration of the Flow

As we observed, our Assumption 1 is equivalent to the assumption that we can compute flows of "frozen" vector fields to arbitrary accuracy. In the case where $M = SO(3)$, as we observed in the introduction, explicit formulas are available to compute the required flows of left invariant vector fields. However, although there are other explicitly integrable geometric objects in this sense (see [28, 29]), their occurrence in practical applications is likely to be limited. It follows that "good" choices of the approximating frame E must be made so that the resulting flows can be calculated off line to arbitrary accuracy with relative efficiency. This seems to be feasible in some cases where M is the tangent space to a Lie group such as $SO(3)$. "Nilpotent" approximating frames might be another choice.

Acknowledgments

This research is supported in part by the NSF grant INT 891-4643 and DMS 910-1964 (PEC) and by NASA grant NAG2-513, NSF grant DMS 910-1089, and the Laboratory for Advanced Computing, (RG).

References

1. M. Austin, P. S. Krishnaprasad, and L.-S. Wang, Symplectic and Almost Poisson Integration of Rigid Body Systems, *Proceedings of the 1991 International Conference on Computational Engineering Science*, ed. S. Atluvi, Melbourne, Australia, August, (1991).
2. J. C. Butcher, An Order Bound for Runge-Kutta Methods, *SIAM J. Numerical Analysis*, Vol. 12, pp. 304–315, (1975).
3. J. C. Butcher, *The Numerical Analysis of Ordinary Differential Equations*, John Wiley, (1986).
4. P. Channell, Symplectic Integration for Particles in Electric and Magnetic Fields, (Accelerator Theory Note, No. AT-6: ATN-86-5). Los Alamos National Laboratory, (1986).
5. P. Channell, and C. Scovel, Symplectic Integration of Hamiltonian Systems, submitted to *Nonlinearity*, June, 1988.
6. A. Chorin, T. J. R. Hughes, J. E. Marsden, and M. McCracken, Product Formulas and Numerical Algorithms, *Comm. Pure and Appl. Math.*, Vol. 31, pp. 205–256, (1978).
7. P. E. Crouch, Spacecraft Altitude Control and Stabilization: Application of Geometric Control to Rigid Body Models, *IEEE Transactions on Automatic Control*, Vol. AC-29, pp. 321–331, (1986).
8. P. E. Crouch, and R. L. Grossman, The Explicit Computation of Integration Algorithms and First Integrals for Ordinary Differential Equations with Polynomial Coefficients Using Trees, *Proceedings of the 1992 International Symposium of Algebraic and Symbolic Computation*, ACM, (1992).
9. P. Crouch, R. Grossman, and R. G. Larson, Computations Involving Differential Operators and Their Actions on Functions, *Proceedings of the 1991 International Symposium on Symbolic and Algebraic Computation*, ACM, (1991).
10. P. Crouch, R. Grossman, and R. Larson, Trees, Bialgebras, and Intrinsic Numerical Integrators, (Laboratory for Advanced Computing Technical Report, # LAC90-R23). University of Illinois at Chicago, May, (1990).

11. R. de Vogelaere, Methods of Integration which Preserve the Contact Transformation Property of the Hamiltonian Equations, Department of Mathematics, University of Notre Dame Report Vol. 4.
12. A. Deprit, Canonical Transformations Depending upon a Small Parameter, *Celestial Mechanics*, Vol. 1, pp. 1–31, (1969).
13. A. J. Dragt, and J. M. Finn, Lie Series and Invariant Functions for Analytic Symplectic Maps, *J. Math. Physics*, Vol. 17, pp. 2215–2227, (1976).
14. K. Feng, The Symplectic Methods for Computation of Hamiltonian Systems, *Springer Lecture Notes in Numerical Methods for P.D.E.'s*, (1987).
15. C. W. Gear, Simultaneous numerical solution of differential-algebraic equations, *IEEE Transactions on Circuit Theory*, Vol. 18, pp. 89–95, (1971).
16. Ge-Zhong, and J. Marsden, Lie-Poisson Hamiltonian-Jacobi Theory and Lie-Poisson Integrations, *Phys. Lett. A.*, Vol. 133, pp. 134–139, (1988).
17. R. Grossman, and R. Larson, The Symbolic Computation of Derivations Using Labeled Trees, *J. Sympolic Computation*, to appear.
18. E. Hairer, C. Lubich, and M. Roche, *The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods*, Springer-Verlag, Berlin, (1989).
19. E. Hairer, S. P. Norsett, and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*, Springer-Verlag, Berlin, (1987).
20. E. Isaacson, and H. B. Keller, *Analysis of Numerical Methods*, Wiley, (1966).
21. A. J. Krener and R. Hermann, "Nonlinear Observability and Controllability," IEEE *Transactions on Automatic Control* A.C.-22, pp. 728–740, (1977).
22. G. Meyer, On the Use of Euler's Theorem on Rotations for the Synthesis of Attitude Control Systems, (NASA Technical Note, NASA TN.D-3643). Ames Research Center, Moffet Field, California, (1966).
23. L. R. Petzold, Order Results for Implicit Runge-Kutta Methods Applied to Differential/Algebraic Systems, *SIAM Journal of Numerical Analysis*, Vol. 23, pp. 837–852, (1986).
24. W. C. Rheinboldt, Differential-Algebraic Systems as Differential Equations on Manifolds, *Mathematics of Computation*, Vol. 43, pp. 473–482, (1984).
25. R. Ruth, A Canonical Integration Technique, *IEEE Transactions on Nuclear Science*, Vol. 30, p. 2669, (1983).
26. J. M. Sanz-Serra, Runge-Kutta Schemes for Hamiltonian Systems, *Bit*, Vol. 28, pp. 877–883, (1988).
27. C. Scovel, Symplectic Numerical Integration of Hamiltonian Systems, *Proceedings of the MSRI Workshop on the Geometry of Hamiltonian Systems*, to appear.
28. F. Silva Leite, and J. Vitoria, Generalization of the De Moivre Formulas for Quaternions and Octonions, *Estudos de Matematica in the Honour of Luis de Albequerque*, Universidade de Coimbra, pp. 121–133, (1991).
29. F. Silva Leite, C. Simoes, and J. Vitoria, Hypercomplex Numbers and Rotations, Universidade Do Minho, 4–8, Vol. 3, pp. 636–643, (1987).
30. Dao-Liu Wang, "Symplectic Difference Schemes for Hamiltonian Systems on Poisson Manifolds," Academia Senica, Computing Center, Beijing, China, (1988).