N91-28788

P-25

Using Trees To Compute Approximate Solutions to Ordinary Differential Equations Exactly

ROBERT GROSSMAN

Laboratory for Advanced Computing
Technical Report LAC90-R24

Department of Mathematics, Statistics,
and Computer Science
University of Illinois at Chicago (M/C 249)
P. O. Box 4348
Chicago, IL 60680

October, 1990

1 Introduction

This extended abstract further develops the algorithms in [8] and [9] for rewriting expressions involving differential operators. The differential operators that we have in mind arise in the local analysis of nonlinear dynamical systems. In this work, we extend these algorithms in two different directions:

1. We generalize the algorithms so that they apply to differential operators on groups. This generalization is important for applications. For example, the nonlinear system describing a robotic joint or a satellite evolves on the group $G = SO(3)$ of spatial rotations. The local study of such systems requires the computation of expressions consisting of differential operators on G.

2. We develop the data structures and algorithms to compute symbolically the action of differential operators on functions. Again, this is crucial for applications. We illustrate this by deriving conditions for a numerical algorithm to remain constrained to a group. In other words, if $x_{n+1} = T(x_n)$ is the update rule for a numerical algorithm evolving on a group G, we would like to choose T so that $x_n \in G$ implies $x_{n+1} \in G$.

For a further discussion of applications of these algorithms, see [5] and [6] and the references given there.

Here is the

Set Up.

1. Let k denote either the real or complex numbers.

2. Let G denote a finite dimensional Lie group over k, g denote its Lie algebra, and $Y_1, \ldots, Y_N$ a basis for g of left-invariant vector fields.

3. Let $R = C^\infty(G)$ denote the algebra of smooth functions on G taking values in k.

4. Fix M derivations of R of the form

$$F_j = \sum_{\mu=1}^{N} a_j^\mu Y_\mu, \quad a_j^\mu \in R, \quad j = 1, \ldots, M, \tag{1}$$

and let A denote the free associative algebra $k<F_1, \ldots, F_M>$ of differential operators generated by $F_1, \ldots, F_M$, with coefficients from k.

We are concerned with the following

Problem. Given a differential operator $p \in A$ and a function $f \in R$, substitute the Equations (1) and compute $p \cdot f$ using as few operations as possible. This problem is interesting since in many cases cancellations take place.

Example 1. Let $G = \mathbf{R}^N$ denote the abelian group,

$$Y_j = \frac{\partial}{\partial x_j}, \quad j = 1, \ldots, N,$$

the (left invariant) coordinate vector fields, and F_1, F_2, F_3 three fixed vector fields defined in terms of the Y_μ via Equations (1). Then the naive substitution of (1) and simplification of $p \cdot f$, where

$$p = F_3F_2F_1 - F_3F_1F_2 - F_2F_1F_3 + F_1F_2F_3 \in A, \quad f \in R,$$

yields $24N^3$ terms, while more specialized algorithms need only compute the $6N^3$ terms which don't cancel. These types of examples are considered in [8] and [9].

Example 2. Consider the local analysis of a nonlinear system of the form

$$\dot{x}(t) = F(x(t)), \quad x(0) = x^0 \in G, \tag{2}$$

where

$$F = \sum_{j=1}^{M} u_j F_j.$$

In practice, the u_j are constants, functions of time, or perturbation parameters. The study of this system typically involves the computations of various series in the algebra A of differential operators. For example, the local flow of the system is determined by the Taylor series

$$\exp hF = 1 + hF + \frac{h^2}{2!}F^2 + \frac{h^3}{3!}F^3 + \ldots \in A[[h]].$$

An alternative to computing higher derivatives F^k is to choose constants c_i, c_{ij}, $i = 1, \ldots, k$, $j < i$, so that the expression

$$\exp hc_k\bar{F}_k \cdots \exp hc_1\bar{F}_1,$$

where

$$\bar{F}_1 = \sum_{\mu=1}^{N} a^\mu(x^0)Y_\mu \in g$$

$$\bar{F}_2 = \sum_{\mu=1}^{N} a^\mu(\exp(hc_{21}\bar{F}_1)\cdot x^0)Y_\mu \in g$$

$$\bar{F}_3 = \sum_{\mu=1}^{N} a^\mu(\exp(hc_{32}\bar{F}_2)\cdot \exp(hc_{31}\bar{F}_1)\cdot x^0)Y_\mu \in g,$$

$$\vdots$$

is equal to $\exp hF$ to order k. Notice that the left invariant vector fields $\bar{F}_j$ arise by "freezing the coefficients" of F at various points along its flow.

Expanding these expressions around the common base point $x^0 \in G$ yields many terms, which must cancel in the end if the algorithm is going to approximate the flow of the underlying nonlinear system. The action of the differential operators $\bar{F}_j$ on the coefficient functions a_k^μ must also be computed. Notice, that unlike Example 1, the Y_μ here do not commute. This example will be considered in more detail in Section 4.

The computations in both examples are easily kept track of by using finite rooted trees, labeled with the symbols $F_1, \ldots, F_M$. It turns out the the vector space, with basis the set of such trees, has an algebraic structure B which is crucial to efficiently organizing the computation. The advantage of working with the trees B is that many terms which cancel in the end need not be computed. See [6] for an expository treatment of this idea. The key observation required for this work is that it is possible to define an action of the algebra B of finite rooted trees, labeled with $F_1, \ldots, F_M$, on the ring of functions R which enjoys essentially all the properties of the familiar action of the algebra A of differential operators generated by $F_1, \ldots, F_M$ on R. It turns out that B is a Hopf algebra, just as A is, and that both actions give R the structure of what is called an H-module algebra.

In Section 2, we review the relevant material from algebra. This material may be skimmed on a first reading. In Section 3, we define the Hopf algebra of Cayley trees and its action on the ring of functions R. In Section 4, we continue the discussion of Example 2.

2 H-module algebras

In this section we review the basic facts about bialgebras and H-module algebras which will be used in the remainder of this paper.

In this section, k can be any field of characteristic 0. By an *algebra* we mean a vector space A over the field k with an associative multiplication and unit. The multiplication can be represented by a linear map $\mu : A \otimes_k A \longrightarrow A$; the unit can be represented by a linear map $k \longrightarrow A$ (the map sending $1 \in k$ to $1 \in A$). The facts that the multiplication is associative, and that $1 \in A$ is a unit, can be expressed by the commutativity of certain diagrams. For example, the commutativity of the diagram

$$\begin{array}{ccc} A \otimes_k A \otimes_k A & \longrightarrow & A \otimes_k A \\ \downarrow & & \downarrow \\ A \otimes_k A & \longrightarrow & A \end{array}$$

where the upper horizontal arrow is the map $\mu \otimes I$, the left vertical arrow is the map $I \otimes \mu$, and the remaining two arrows are the map μ, expresses the associativity of multiplication.

The dual notion to an algebra is a *coalgebra*: a vector space C over the field k together with a coassociative coproduct $\Delta : C \longrightarrow C \otimes_k C$ and a counit $\epsilon : C \longrightarrow k$. The fact that Δ is coassociative and that ϵ is a counit is expressed by diagrams which are dual to the diagrams which express the facts that the multiplication of an algebra is associative, and that $1 \in A$ is a unit: they are the same diagrams, with the direction of all arrows reversed. For example, coassociativity is expressed by the commutativity of the diagram

$$\begin{array}{ccc} C \otimes_k C \otimes_k C & \longleftarrow & C \otimes_k C \\ \uparrow & & \uparrow \\ C \otimes_k C & \longleftarrow & C \end{array}$$

where the upper horizontal arrow is the map $\Delta \otimes I$, the left vertical arrow is the map $I \otimes \Delta$, and the remaining two arrows are the map Δ. Often the element $\Delta(c) \in C \otimes_k C$ is written $\sum_{(c)} c_{(1)} \otimes c_{(2)}$.

A *bialgebra* is a vector space H over k which has both an algebra and a coalgebra structure, such that the coalgebra structure maps are algebra homomorphisms, or equivalently, the algebra structure maps are coalgebra homomorphisms. (This equivalence can be seen by expressing the assertion that the coalgebra structure maps are algebra homomorphisms as a set of commutative diagrams: this set of diagrams is self-dual.)

Some examples of bialgebras are the following:

1. Let G be a group, and let kG be the group algebra of G: the vector space kG has the elements of G as a basis, with multiplication defined by extending the multiplication on G linearly. The coproduct and counit of kG are defined by

$$\left.\begin{array}{rcl} \Delta(g) & = & g \otimes g \\ \epsilon(g) & = & 1 \end{array}\right\} \qquad g \in G.$$

2. Let G be an affine algebraic group, and let $k[G]$ be the algebra of representative functions on G. The algebra structure of $k[G]$ is the usual algebra structure of functions with point-wise multiplications. The coproduct arises from the group multiplication $G \times G \to G$, which induces the map $k[G] \to k[G \times G] \cong k[G] \otimes_k k[G]$. The counit arises from the map $\{e\} \to G$, where $\{e\}$ is the single-element group.

3. Let L be a Lie algebra over k, and let $U(L)$ be the universal enveloping algebra of L. The coproduct and counit of $U(L)$ are defined by

$$\left.\begin{array}{rcl} \Delta(x) & = & 1 \otimes x + x \otimes 1 \\ \epsilon(x) & = & 0 \end{array}\right\} \qquad x \in L,$$

and extended to all of $U(L)$ using the fact that Δ and ϵ are algebra homomorphisms.

Usually, in studying bialgebras, an additional condition is imposed which is analogous to the assertion that a semigroup is a group. Such bialgebras are called *Hopf algebras*. The bialgebras which we consider in this paper (such as the universal enveloping algebra of a Lie algebra) satisfy this condition automatically.

A coalgebra is said to be *cocommutative* if it satisfies $\Delta = T \circ \Delta$, where T is the map $T : C \otimes_k C \to C \otimes_k C$ defined by $T(x \otimes y) = y \otimes x$. Note that the bialgebras in Examples 1 and 3 are cocommutative.

A vector space V over k is said to be *graded* if there is a sequence of subspaces $V_0, V_1, \ldots$ such that

$$V \cong \bigoplus_{n=0}^{\infty} V_n.$$

A graded vector space V is said to be *connected* if $V_0 \cong k$.

Let H be a bialgebra. A *H-module algebra* is an algebra R which is an H-module such that the action satisfies

$$h \cdot (fg) = \sum_{(h)} (h_{(1)} \cdot f)(h_{(2)} \cdot g), \qquad \text{for all } h \in H, \quad f, g \in R.$$

Remark 2.1 If $g \in H$ satisfies $\Delta(g) = g \otimes g$ and R is an H-module algebra, then g acts as an endomorphism of R; if $x \in H$ satisfies $\Delta(x) = 1 \otimes x + x \otimes 1$ and R is an H-module algebra, then x acts as a derivation of R.

3 H-module algebras and Cayley trees

In this section we describe a bialgebra structure on the vector space with basis all equivalence classes of rooted trees. The relation between trees and differential operators goes back at least as far as Cayley [3] and [4]. Important use of this relation has been made by Butcher in his work on higher order Runge-Kutta algorithms [1] and [2]. In this section and the next, we follow the treatment in [8] and [9]. By a tree we mean a nonempty finite rooted tree, and by a forest we mean a finite family of finite rooted trees, possibly empty.

Suppose $\{F_1, \ldots, F_M\}$ is a set of formal symbols (which later will be the names of differential operators). By a *labeled tree* we mean a tree for which we have assigned an element of $\{F_1, \ldots, F_M\}$ to each node, other than the root, of the tree. We say that a tree is *ordered* in case there is a partial ordering on the nodes such that the children of each node are non-decreasing with respect to the ordering.

We now describe the bialgebra structure on spaces of trees. Let

$$k\{\mathcal{T}(F_1, \ldots, F_M)\}$$

denote the vector space which has as basis all equivalence classes of labeled, ordered trees. The vector space $k\{\mathcal{T}(F_1, \ldots, F_M)\}$ is graded, with the grading given as follows: if the tree t has $n+1$ nodes, then

$$t \in k\{\mathcal{T}(F_1, \ldots, F_M)\}_n.$$

We now define the multiplication on $k\{\mathcal{T}(F_1, \ldots, F_M)\}$. Since the set of labeled, ordered trees form a basis for $k\{\mathcal{T}(F_1, \ldots, F_M)\}$, it is sufficient to describe the product of two such trees. Suppose that t_1 and t_2 are labeled, ordered trees. Let $s_1, \ldots, s_r$ be the children of the root of t_1. If t_2 has $n+1$ nodes (counting the root), there are $(n+1)^r$ ways to attach the r subtrees of t_1 which have $s_1, \ldots, s_r$ as roots to the labeled tree t_2 by making each s_i the child of some node of t_2, keeping all the original labels. Order the nodes in the product so that the nodes which originally belonged to each tree retain the same relative order to each other, but all the nodes that orginally belonged to t_1 are greater in the ordering than the nodes that originally belonged to t_2. The product $t_1 t_2$ is defined to be the sum of these $(n+1)^r$ labeled trees. It can be shown that this product is associative, and that the trivial labeled tree consisting only of the (unlabeled) root is a right and left unit for this product. For details, see [7].

We now define the comultiplication on $k\{\mathcal{T}(F_1, \ldots, F_M)\}$. If t is a tree whose root has children $s_1, \ldots, s_r$, the coproduct $\Delta(t)$ is the sum of the 2^r terms $t_1 \otimes t_2$, where the children of the root of t_1 and the children of the root of t_2 range over all 2^r possible partitions of the children of the root of t into two subsets. The labels remain the same, and the ordering is handled in the same way as in the product. The map ϵ which sends the trivial labeled tree to 1 and every other tree to 0 is a counit for this coproduct. In [7], it is shown that these algebra and coalgebra structures are compatible, proving the

Theorem 3.1 *The space $k\{\mathcal{T}(F_1, \ldots, F_M)\}$ is a graded connected cocommutative bialgebra.*

We call this algebra the algebra of *Cayley trees*.

We now define an action of the algebra of Cayley trees

$$B = k\{\mathcal{T}(F_1, \ldots, F_M)\}$$

on the ring R, making R a B-module algebra, which captures the action of trees as higher derivations. The action is defined using the map

$$\psi : k\{\mathcal{T}(F_1, \ldots, F_M)\} \rightarrow \mathrm{End}_k\, R,$$

as follows:

1. Given a labeled, ordered tree t with $m+1$ nodes, assign the root the number 0 and assign the remaining nodes the numbers $1, \ldots, m$. We identify the node with the number assigned to it. To the node k associate the summation index μ_k. Denote $(\mu_1, \ldots, \mu_m)$ by μ.

2. For the labeled tree t, let k be a node of t, labeled with F_{γ_k} if $k > 0$, and let $l, \ldots, l'$ be the children of k. Define

$$\begin{aligned} R(k;\mu) &= Y_{\mu_l} \cdots Y_{\mu_{l'}} a_{\gamma_k}^{\mu_k}, \quad \text{if } k>0 \text{ is not the root;} \\ &= Y_{\mu_l} \cdots Y_{\mu_{l'}}, \quad \text{if } k=0 \text{ is the root.} \end{aligned}$$

Note that if $k > 0$, then $R(k;\mu) \in R$.

3. Define

$$\psi(t) = \sum_{\mu_1,\ldots,\mu_m=1}^{N} R(m;\mu)\cdots R(1;\mu)c(0;\mu).$$

4. Extend ψ to all of $k\{\mathcal{T}(F_1, \ldots, F_M)\}$ by linearity.

It is straightforward to check that this action of B on R makes R into a B-module algebra.

We summarize with the following theorem.

Theorem 3.2 *Let G denote a finite dimensional Lie group and R the algebra of smooth functions on G, as detailed in the Set Up. Let B denote the algebra of Cayley trees $k\{\mathcal{T}(F_1, \ldots, F_M)\}$. Then R is a B-module algebra with respect to the action defined by ψ.*

Remark 3.1 The standard action of the algebra A of differential operators generated by $F_1, \ldots, F_M$ on the algebra of smooth functions R gives R the structure of a A-module algebra. It is easy to relate these two H-module algebra structures on R and this observation is the basis for our algorithms.

Let

$$\phi : A \longrightarrow B$$

denote the map sending the generator F_j of the algebra A to the tree consisting of two nodes: the root and a single child labeled F_j. Extend ϕ to be an algebra homomorphism. Let χ denote the map

$$A \rightarrow \mathrm{End}_k R$$

defined by using the substitution (1) and simplifying to obtain an endomorphim of R.

Theorem 3.3 *(i) The maps χ, ϕ and ψ are related by $\chi = \psi \circ \phi$. (ii) Fix a function $f \in R$ and a differential operator $p \in A$. Then*

$$p \cdot f = \phi(p) \cdot f.$$

Here the action on the left views R as an A-module algebra, while the action on the right views R as B-module algebra.

The first assertion is proved in [9] and the second assertion follows from the first assertion and the definitions.

Using this theorem, it is easy to give an algorithm to solve the Problem posed in Section 1. We defer to later paper a complete analysis of the complexity of the algorithm and simply remark here that in many examples the algorithm results in a savings which is exponential in the degree of the differential operator.

Algorithm. Given a smooth function $f \in R$ and a differential operator $p \in A$, compute the function $p \cdot f$ via $\phi(p) \cdot f$.

4 Applications

We use the notation of the Set Up from Section 1. Let $\exp(hF)x$ denote the resulting of flowing for time h along the trajectory of the nonlinear system (2) through the initial point $x^0 \in G$. We require a theorem concerned with the explicit computation of terms in the Taylor series expansion of a solution of (2). This is one of the main applications of the symbolic calculus described in the sections above.

This theorem is most easily stated if we introduce two additional operations on the algebra of Cayley trees B. Given $\alpha, \beta \in B$, define the *meld product* $\beta \odot \alpha$ to be the labeled, ordered tree obtained by identifying the roots of the two trees. The meld product is then extended to all of B by linearity. Given a derivation $F \in \mathrm{Der}(R)$, let β be the tree $\phi(F)$ and let $\alpha \in B$. Recall β is a tree consisting of a root and a node labeled F. We define the *composition product* $\beta \circ \alpha$ to be the tree formed by attaching the subtrees whose roots are the children of the root of α to the node labeled F of the tree β. If $\alpha \in B$ is a tree, define the *exponential* and *Meld-exponential* of a tree by the formal power series

$$\exp(h\alpha) = 1 + h\alpha + \frac{h^2}{2!}\alpha^2 + \frac{h^3}{3!}\alpha^3 + \cdots$$

$$\mathrm{Mexp}(h\alpha) = 1 + h\alpha + \frac{h^2}{2!}\alpha \odot \alpha + \frac{h^3}{3!}\alpha \odot \alpha \odot \alpha + \cdots.$$

Theorem 4.1 *(i) Assume $f \in R$ and $F \in \mathrm{Der}(R)$. If f is analytic near x, then for sufficiently small h,*

$$f(\exp(hF)x) = \exp(h\phi(F)) \cdot f|_x.$$

(ii) Let $F = \sum_{\mu=1}^{N} a^{\mu}(\exp(hG)x^0)Y_{\mu}$, *where* $G \in \mathrm{Der}(R)$, *and* $x^0 \in G$. *Then*

$$F \cdot f = (\phi(F) \circ \mathrm{Mexp}(hG)) \cdot f.$$

Using this theorem, it is easy to analyze the numerical algorithm described in Example 2 of Section 1. For typographical reasons, we use the following one dimensional notation for trees[1]: the tree consisting of a root and a single child labeled F_1 is denoted $I[F_1]$; the tree consisting of a root and two children labeled F_1 and F_2 is denoted $I[F_1, F_2]$; the tree consisting of a root, with a single child labeled F_1, which itself has two children labeled F_2 and F_3 is denoted $I[F_1[F_2, F_3]]$, etc. Note that the labels need not be distinct, but their order is important.

Consider the expression

$$\exp hc_3 F_3 \exp hc_2 F_2 \exp hc_1 F_1$$

computed to order h^3. Let $p \in A$ denote the resulting expression. The image $\phi(p) \in B$ contains the following terms:

$$\frac{h^3 c_2 c_{21}^2}{2!} I[F[F,F]] + \frac{h^3 c_3 c_{31}^2}{2!} I[F[F,F]] + \frac{h^3 c_3 c_{32}^2}{2!} I[F[F,F]]$$
$$+ h^3 c_3 c_{31} c_{32} I[F[F,F]].$$

Our goal is to choose the constants c_i and c_{ij} so that that

$$\exp hF = p + O(h^4).$$

One of the third order term arising from $\phi(\exp hF)$ is $\frac{h^3}{3!} I[F, [F, F]]$. Setting the coefficients of these trees equal to each other yields the constraint:

$$\frac{c_2 c_{21}^2}{2!} + \frac{c_3 c_{31}^2}{2!} + \frac{c_3 c_{32}^2}{2!} + c_3 c_{31} c_{32} = \frac{1}{3!}$$

Other constraints arise from the other trees. We have coded this algorithm in Maple, Mathematica, and Snobol and are currently experimenting with it.

[1]The notation is due to Peter Olver, as is some of the Mathematica code used to generate these examples.

References

[1] J. C. Butcher, "An order bound for Runge-Kutta methods," *SIAM J. Numerical Analysis*, **12** (1975), pp. 304–315.

[2] J. C. Butcher, *The Numerical Analysis of Ordinary Differential Equations*, John Wiley, 1986.

[3] A. Cayley, "On the theory of the analytical forms called trees," in *Collected Mathematical Papers of Arthur Cayley*, Cambridge Univ. Press, Cambridge, 1890, Vol. 3, pp. 242–246.

[4] A. Cayley, "On the analytical forms called trees. Second part," in *Collected Mathematical Papers of Arthur Cayley*, Cambridge Univ. Press, Cambridge, 1891, Vol. 4, pp. 112–115.

[5] R. Grossman, "The evaluation of expresssions involving higher order derivations," *Journal of Mathematical Systems, Estimation, and Control*, Vol. 1, 1990.

[6] R. Grossman, "Using trees to compute approximate solutions of ordinary differential equations exactly," *Computer Algebra and Differential Equations*, M. F. Singer, editor, Academic Press, New York, 1991, in press.

[7] R. Grossman and R. Larson, "Hopf algebraic structures of families of trees," *J. Algebra*, Vol. 26 (1989), pp. 184–210.

[8] R. Grossman and R. Larson, "Labeled trees and the efficient computation of derivations," in *Proceedings of 1989 International Symposium on Symbolic and Algebraic Computation*, ACM, 1989, pp. 74-80.

[9] R. Grossman and R. Larson, "The symbolic computation of derivations using labeled trees," *Journal of Symbolic Computation*, to appear.