Computer Algebra and Differential Equations — An Overview

Werner M. Seiler
Institut für Algorithmen und Kognitive Systeme
Universität Karlsruhe, 76128 Karlsruhe, Germany
werner.seiler@informatik.uni-karlsruhe.de

We present an informal overview of approaches to differential equations popular in computer algebra. We also give a brief outlook on a MuPAD environment for differential equations currently under construction.

Introduction

Differential equations represent one of the largest fields within mathematics. Besides being an interesting subject of their own right one can hardly overestimate their importance for applications. They appear in natural and engineering sciences and increasingly often in economics and social sciences. Whenever a continuous process is modeled mathematically, chances are high that differential equations are used.

Thus it is not surprising that differential equations also play an important role in computer algebra and most general purpose computer algebra systems provide some kind of `solve` command. Many casual users believe that designing and improving such procedures is a central problem in computer algebra. But the real situation is somewhat different. Many computer algebra applications to differential equations work indirectly; they help to study and understand properties of the solution space.

The purpose of this article is to sketch in an informal way some of the main research directions in this field. This will be done without any mathematical details. For readers who want to know more many references are given. We have chosen them mainly so that they can serve as a good starting point for deeper study; thus often introductory articles or books have been chosen and not the historically first or the most "ground breaking" work.

As a further source of references one should also mention the excellent survey [78] by Singer. It gives much more details, especially on the more algebraic approaches, and contains a large bibliography. The same holds for the more specialized surveys of Hereman [37, 38] covering symmetry theory and related fields and the one of MacCallum [50] on the integration of ordinary differential equations.

Almost any constructive method for differential equations has meanwhile been implemented in some computer algebra system. One can, however, distinguish certain approaches which have found most attention (at least measured in the number of articles devoted to them). We will discuss later the following five fields: (i) *symmetry analysis*, (ii) *singularity analysis*, (iii) *completion*, (iv) *differential ideal theory*, and (v) *differential Galois theory*.

A comparison of the impact made by symmetry analysis and by differential Galois theory, respectively, demonstrates the importance of computer algebra tools. The latter one is a hardly known theory studied by a few pure mathematicians. The former one remained in the same state for many decades following Lie's original work. One reason was definitely the tedious determination of the symmetry algebra. As soon as computer algebra systems emerged, the first packages to set up at least the determining equations were developed. Since then Lie methods belong to the standard tools for treating differential equations and are used by many applied mathematicians and by physicists.

Solving Differential Equations

The `solve` commands provided by most computer algebra systems for differential equations mainly apply some standard techniques like those listed in Zwillinger's handbook [94] or they may even try some "pattern matching" in a collection of solved equations like Kamke [44]. Thus they can treat only certain classes of differential equations, but heuristics often extend their applicability. A typical task for heuristics is to find a transformation of the given differential equation such that it can be handled by the implemented methods.

Although this approach solves more differential equations than one might expect (see e. g. the recent review by Postel and Zimmermann [63]), it has some drawbacks. A major one is that no information is obtained, if the `solve` command does *not* return a solution. It could be that the given differential equation has indeed no solution (or at last none in closed form) or that simply the heuristics were not able to determine a suitable transformation.

For that reason researchers in computer algebra are more interested in *decision algorithms*. These either yield a solution in a specific class of functions or decide that no such solution exists. However, so far only for linear ordinary differential equations such algorithms are known. There it is possible to decide with the help of differential Galois theory whether or not Liouvillian solutions exist (see below).

There exists a number of reasons for this perhaps disappointing situation. First of all, computability theory yields principal limits to what can be solved [19], i. e. there exist differential equations where one can prove that it is not possible to construct algorithmically the solution. Then, ideally the algorithm should return the *general solution*. But for nonlinear equations it is surprisingly difficult even just to define this term. A resolution of this problem based on differential ideal theory (see below) was only recently presented [41].

Intuitively one would expect that the general solution depends on some arbitrary parameters (constants or functions) and every solution of the differential equation can be obtained by a suitable specialization of these. This works fine for linear equations where the solution space has the structure of a vector space. But many nonlinear equations possess in addition *singular integrals* not contained in the general solution. They are either envelopes or asymptotics of elements in the general solution.

Similarly, defining the term *closed form solution* is notoriously difficult. Is a solution in terms of, say, Bessel functions in closed form or not? Up to now no generally accepted definition has emerged. Loosely spoken the basic idea behind "closed form" is that of finite constructibility out of a set of "elementary functions". One large class that comprises most of the expressions one would usually consider as closed form is the class of Liouvillian functions which will be discussed below.

On the practical side one must see that even if a solution in closed form can be computed it may take very long and the result may be completely useless, as it is too large. Especially, if the main goal is to obtain an impression of the behavior of the solution, it is usually much more efficient to resort to numerical methods. For that reason many computer algebra systems provide at least for ordinary differential equations some standard numerical integrators like Runge-Kutta methods etc.

In any case one can state that a notable solution theory exists only for ordinary differential equations (see e. g. the survey [50]). As we will see later in the section on differential Galois theory, algorithms to compute the general solution suffer from a very high complexity and are in practice often rather useless, especially for higher order equations. One way out is to incorporate heuristics as mentioned above.

Another possibility that also addresses the problem of useless output is to aim from the very beginning only for "simple" solutions [5, 9]. Popular variants are polynomial, rational or exponential solutions. Because of their simple structure it is often possible to determine such solutions, if they exist, rather efficiently. Obviously, this yields only in special cases the general solution.

For partial differential equations we are still far away from any general solution theory. If a computer algebra system claims that it can solve partial differential equations, this is usually in some sense a "cheat". Almost always it means nothing else than that the system knows a bit of the theory of characteristics which reduces quasi-linear first order equations to systems of ordinary differential equations. As the latter ones are in general non-linear it is still a formidable task to solve them, but for some classical partial differential equations like the wave equation it is fairly simple.

At the end of the last century mathematicians designed some solution methods for partial differential equations [88]. However, most of them are meanwhile almost forgotten; at least they are no longer found in textbooks on differential equations. It could be quite interesting to revive some of them for use in computer algebra systems.

Symmetry Analysis

Symmetry analysis has made the strongest impact on computer algebra applications to differential equations. The most general definition of a symmetry is that of a transformation that maps solutions into solutions. Depending on the kind of transformations considered one obtains different kinds of symmetries. One possible application of symmetries is the construction of (special) solutions. Other goals are classifications, a proof of complete integrability, separation ansätze, conservation laws and much more. Meanwhile several excellent textbooks on this subject are available, e. g. [7, 56, 83].

Symmetry analysis goes back to the seminal work of Lie. He developed the concept of Lie groups in his quest for a Galois theory for differential equations. As we will see later, not much has remained of this original motivation. Symmetry and Galois theory have developed in very different directions. Even the relation between the Lie symmetry and the Galois group of a differential equation is rather unclear.

The most popular variant of symmetry analysis deals with *Lie point symmetries*. They are generated by vector fields acting on the space of independent and dependent variables. These vector fields span the Lie algebra of the Lie group of symmetries. The decisive observation of Lie was that for most purposes it suffices to work with the vector fields (or infinitesimal symmetries) instead of the symmetries themselves. This leads effectively to a linearization of the problem.

The symmetry generators arise as the solutions of a linear system of partial differential equations, the *determining system*. For ordinary differential equations it is unfortunately often as difficult to solve this system as to solve the original one. This holds especially for first order equations where the original equation is just the characteristic equation of the determining equation. For partial differential equations the determining system is typically very over-determined and contains often some trivial equations allowing in many cases a rather straightforward solution.

For ordinary differential equations the existence of a sufficiently large, *solvable* symmetry algebra implies that its general solution can be constructed by quadratures only, as each symmetry allows us to reduce the order of the equation by one. In the case of partial differential equations symmetry reductions yield only special solutions, namely those being invariant under the symmetry group. Here each symmetry allows us to reduce the number of independent variables by one.

However, at intermediate steps of the reduction again linear partial differential equations must be solved. For in order to obtain the reduction, one must either perform a coordinate transformation such that the symmetry generator is rectified (so-called *canonical coordinates*) or the *differential invariants* of the symmetry must be determined. These are functions annihilated by the generator.

Thus the usefulness of Lie symmetries depends crucially on the ability to solve effectively all the arising linear partial differential equations. At first sight it might look, as if, especially for ordinary differential

equations, we made the problem only worse. But in many cases of practical interest it turns out that is much simpler to solve these linear partial differential equations instead of the original equation.

There exist so many implementations of symmetry methods that it is rather difficult to keep an overview; we refer again to the surveys by Hereman [37, 38]. In almost any computer algebra system one can find a package for setting up the determining system. A few of the packages (e. g. [36, 72]) try furthermore some heuristics to solve it automatically. Again it is rather surprising how often this suffices to obtain the complete symmetry algebra. The symmetry package of MAPLE [13] is somewhat unusual, as it uses the exterior systems approach of Harrison and Estabrook [33].

Although Lie point symmetries proved to be very useful in many applications, many differential equations of practical interest have no such symmetries. There are two basic methods to generalize the approach. One can consider more general transformations; this leads to *generalized* or *Lie-Bäcklund symmetries* [2]. Alternatively, one weakens the requirement that every solution is mapped into a solution; this yields the so-called *non-classical methods*. In both cases the explicit construction of the symmetries becomes considerably more difficult.

Generalized symmetries are especially of interest for completely integrable systems [25, 91]. The existence of a *recursion operator* or a *master symmetry* generating an infinite hierarchy of symmetries is a strong indication that the considered system is completely integrable. Reduction with respect to generalized symmetries is an important tool for the construction of *soliton solutions*. It is also possible to classify nonlinear partial differential equations using these symmetries [54]. Some *MuPAD* packages for symmetries of integrable systems are described in [26].

Non-classical reductions can be understood best within the general scheme of augmenting a given differential equation with *differential constraints* [57]. This corresponds to requiring that only some solutions are mapped into solutions, therefore one hopes to find more symmetries (these are sometimes called *weak symmetries*). In this approach the emphasis lies less on group theory but on the theory of over-determined systems of partial differential equations and thus on questions of completion (see below and [76]).

The first non-classical method was developed by Bluman and Cole [6]. They added the invariant surface condition as constraint. Although this leads for many differential equations to new reductions, the drawback is that the determining system becomes nonlinear. The *direct method* of Clarkson and Kruskal [15] tries to reduce a given partial differential equation to a system of ordinary differential equations by constructing a good ansatz; it corresponds to a special case of the method of Bluman and Cole.

The main problem in the method of differential constraints is to find compatible constraints leading to non-trivial reductions. Besides using the invariant surface condition no systematic way has been discovered so far and thus it remains essentially a game of "try and error". For this reason differential constraints have not yet found much attention in applications.

Singularity Analysis

This field splits into several directions depending on what kind of singularities one is interested in. One important direction is the *Painlevé theory* [42]. It is based on complex analysis and was introduced by Painlevé while searching for new special functions. There still exists a strong connection between the Painlevé theory of ordinary differential equations and special function theory.

Painlevé tried to classify all second order ordinary differential equations where the solutions have at most poles as *movable* singularities. Movable means here that the location of the singularity depends on the initial data. A generalization of this idea to partial differential equations was later given by Weiss *et al.* [90]. Here a whole singularity manifold must be considered.

If all singularities are poles, no branch points appear in the (general) solution and it is single valued. A differential equation without movable branch points is said to possess the *Painlevé property* or to be integrable in the sense of Painlevé. In general, it is not possible to check algorithmically whether or not a given differential equation has the Painlevé property. But there exist methods to check at least some necessary conditions; such methods are usually call *Painlevé test* [16].

In these methods one usually tries to construct a Laurent series around the singularity. Essentially, the test is passed, if this expansion has sufficiently many *resonances* or *Fuchsian indices* (free coefficients) to represent the general solution and if these are non-negative. In the case of negative resonances a peturbation approach [17] yields further information. The tests are only conclusive, if they fail, as the checked conditions are not sufficient for the Painlevé property.

The Painlevé approach is very popular in the theory of completely integrable systems, as the Painlevé test represents an important indicator for complete integrability and can be checked comparatively easily. The *Painlevé conjecture* states that every ordinary differential equation obtained as a symmetry reduction of a completely integrable system is of Painlevé type. So far only weakened versions of it have been proved [1, 53]. Truncated Painlevé expansions are useful for the construction of Bäcklund transformations, Lax pairs and much more [89]. There exist also relations to non-classical symmetry reductions [23].

A very general REDUCE implementation of various forms of the Painlevé test for ordinary differential equations was recently presented by Scheen [70] (together with a brief review of other implementations). A MACSYMA implementation for partial differential equations is due to Hereman and Van den Bulck [39].

Another form of singularity analysis deals with linear differential operators with polynomial coefficients. Here one is interested in *fixed* singularities of the solutions. They can be located only at the zeros of the leading coefficient of the operator. Using the *Newton polygon* of the operator they are classified into *regular* and *irregular* ones. A brief introduction into the theory can be found in [18] (see also [20]).

The goal of the theory is to construct formal power series solutions in the neighborhood of a singularity. Depending on the character of the singularity different ansätze must be used. In the case of an irregular singularity the most difficult part is to determine the *exponential part* of the solution and its *ramification index*. All this can be done by analyzing the Newton polygon.

There exist various algorithms for the construction of the series, partly dating back to Frobenius. Some of them have been implemented in the MAPLE package DESIR [59]. A main problem in the concrete application is that one cannot use an approximation of the location of the singularities. Thus one must not only solve polynomial equations but in general work with algebraic numbers which is quite expensive in any computer algebra system.

Recent work concerns an extension of the theory to first order systems [4]. In principle, one can transform any system into a single equation of higher order; traditionally this is done using *cyclic vectors*. However, this approach is rather inefficient, as many artificial singularities may appear. Therefore one is interested in dealing directly with the system. *Moser's algorithm* classifies the singularities into regular and irregular ones. There exists a rational version of it avoiding the use of algebraic extensions [3].

Completion

Most textbooks on differential equations treat only *normal* systems (or systems in *Cauchy-Kowalevsky form*). For ordinary differential equations this implies that one always assumes that the equations can be solved for the highest order derivatives. For partial differential equations one must furthermore assume the existence of a distinguished independent variable such that one can solve for its derivatives to obtain the Cauchy-Kowalevsky form. However, in many fields one encounters systems of differential equations

which are not normal. A simple example are the determining systems appearing in symmetry analysis (see above) which are usually over-determined. Non-normal systems also occur naturally in differential geometry and in theoretical physics (gauge theories).

For a non-normal system it is a priori not clear whether it has any solutions. It may happen that the system is inconsistent. This can only be decided after the construction of all *integrability conditions*. These are differential equations satisfied by any solution of the system but which are nevertheless *algebraically* independent of it. While it makes no problem to construct an integrability condition (typically this requires only taking cross-derivatives), it is not so easy to decide when *all* have been found, as in principle an infinite number of conditions must be checked.

The process of finding all integrability conditions is called *completion* of the differential equation. It results in a *formally integrable system*, as for a system containing all its integrability conditions it is straightforward to construct order by order a formal power series solution. Under some assumptions it is sometimes possible to show the convergence of the series. This leads for analytic equations to existence and uniqueness theorems like the Cartan-Kähler theorem (the well-known Cauchy-Kowalevsky theorem is a special case of it). For non-analytic equations solvability is a much more complicated question due to Lewy type effects [48].

The first systematic approach to the problem of completion was probably provided by the Janet-Riquier theory [43] with the introduction of *passive* systems. Their definition is based on a *ranking* of the derivatives which decides in what order the integrability conditions are constructed. Implementations have been undertaken by several authors, see e. g. [65, 84]. The so-called differential Gröbner bases of Mansfield (see below) may be considered as an extension of this approach.

In geometric theories the notion of a passive system is replaced by *involution*. It combines a geometric definition of formal integrability with an algebraic criterion for the termination of the completion. As an intrinsic concept involution requires no coordinate dependent ingredients like a ranking. Hartley and Tucker [35, 34] implemented in REDUCE the Cartan-Kähler theory [10] for exterior systems. An AXIOM implementation of a completion algorithm in the jet bundle formalism based on the formal theory of Pommaret [61] was presented in [71, 74].

Such completion algorithms can be very useful in the symmetry analysis of differential equations (see above). Once a system is either passive or involutive, one can make statements about the size of the solution space [65, 73]. Thus it is possible to compute the size of the symmetry group without explicitly solving the determining system or to determine the loss of generality in a symmetry reduction [75]. Recently it was shown that it is even possible to algorithmically determine the structure of the symmetry algebra without solving the determining system [49, 66].

There is a close relationship between the concepts discussed here and Gröbner bases in commutative algebra. This holds especially for the Janet-Riquier theory where rankings play a similar role as for the definition of a Gröbner basis. Therefore one sometimes find the term *differential Gröbner basis* for an involutive or passive system. Integrability conditions arising from cross-derivatives may be considered as "differential S-polynomials".

These analogies acquire a precise meaning only in the context of differential algebra (discussed in the next section). One should, however, mention that there is a one-to-one correspondence between linear systems of partial differential equations in one dependent variable and polynomial ideals. This has lead in commutative algebra to the new concept of an *involutive basis* of an ideal [31]. These bases are computed using algorithms coming from the completion theory of differential equations, but they are ordinary (though not reduced) Gröbner bases. It has been shown that in some cases these algorithms are considerably faster than the classical Buchberger algorithm.

Differential Ideal Theory

Like the differential Galois theory discussed in the next section, differential ideal theory belongs to the field of *differential algebra*. It can be informally described as an attempt "to write differential in front of everything in algebra". Thus it deals with differential rings, differential fields etc. Of course, this requires an algebraic definition of differentiation. In differential algebra any mapping that is linear with respect to addition and satisfies the Leibniz or product rule is called a *derivation*. A differential ring is a commutative ring together with one (or more) derivation.

Differential polynomials arise by adjunction of differential indeterminates to a differential ring. But the ring of differential polynomials is not Noetherian. Adjoining a differential indeterminate corresponds to adjoining infinitely many algebraic indeterminates, as one must introduce all its derivatives as additional, algebraically independent variables. Thus Hilbert's Basis Theorem does not apply.

A *differential ideal* is an ideal which is in addition closed under the derivation of the differential ring. Many of the basic ideas in differential ideal theory can be traced back to Ritt [67]; the most advanced book is still the one by Kolchin [46]. Like in the purely algebraic theory *Gröbner bases* or *characteristic sets* are the most important tools. As the ring of differential polynomials is not Noetherian, algorithms along the lines of the Buchberger algorithm do not terminate in general [14]. This is related to the fact that the *ideal membership problem* is undecidable for arbitrary differential ideals [28]. However, this result is more of theoretical interest, as for finitely generated ideals (and that is what one encounters in applications) the decidability is still an open question.

There exist basically two strategies to circumvent this principal problem. One can either restrict the class of ideals the algorithm is supposed to handle or one weakens the properties expected of a differential Gröbner basis. Many of the completion algorithms based on Janet-Riquier theory (see above) can be considered as simple examples for the first strategy. An example for the second one is given by the *differential Gröbner bases* of Mansfield [52]. They use only pseudo-reductions and have thus weaker properties than their algebraic counterpart.

Recently, Boulier *et al.* [8] presented a so-called *Rosenfeld-Gröbner algorithm* which computes a representation for the radical ideal of a finitely generated differential ideal in the following form. The radical is written as a finite intersection of saturations ideals; these are radical differential ideals defined by a system of differential polynomial equations and inequalities. This representation allows for an easy algorithmic test of radical ideal membership and for computing formal power series solutions.

Open problems are to obtain a minimal decomposition, i.e. to use only a minimal number of saturation ideals, and to find bases for these ideals (avoiding the inequalities). These questions are closely related to the inclusion problem for differential ideals which in turn can be seen as the problem of determining the relation between the singular and the general solutions of a differential equation. The principal obstacle in the construction of the bases is a very typical one in differential algebra. A theorem of Ritt asserts that by taking sufficiently many derivatives of the equations one can always get a basis but no bound for the number of derivatives needed is known.

Differential ideal theory is applied in *automatic theorem proving* in differential geometry [93]. This is similar to the use of algebraic ideal theory in theorem proving in elementary geometry. For this kind of applications characteristic sets seem to be more useful than Gröbner bases. A nice example for the possibilities here is the automatic derivation of Newton's law of gravity from the three Kepler laws [92].

Besides ideals of differential polynomials there has also been some work on ideals of linear differential operators or ideals of the Weil algebra [27]. However, here one is dealing with non-commutative rings. In some sense one can also consider the Cartan-Kähler theory mentioned above as a kind of differential ideal theory, as it represents differential equations by closed ideals of differential forms.

Differential Galois Theory

Already Lie was looking for a differential analog of the (algebraic) Galois theory, when he introduced Lie groups. What is nowadays usually called differential Galois theory [79] resembles, however, only faintly his ideas. It considers exclusively linear ordinary differential equations and culminates in the *Singer algorithm* for computing Liouvillian solutions of equations with Liouvillian coefficients [77, 80].

Determining the solutions of linear differential equations is a very classical topic and many famous mathematicians like Liouville, Fuchs, Klein or Jordan studied it in the last century and their results are still very important for the design of algorithms. Differential Galois theory was essentially founded by Picard and Vessiot. It was given its modern form by Kolchin [46]. Pommaret [62] developed an alternative theory following more closely Lie's ideas and using the formal theory.

Liouvillian functions comprise essentially all expressions one can "easily write down". Allowed operations are the usual arithmetic operations, roots, exponentials, logarithms, integrals and algebraic functions. A more formal definition can be given via a tower of simple extensions of the field of rational functions. An important point is that for any Liouvillian function one needs only a finite number of extensions, thus it is algorithmically constructible. Most expressions one would call "closed-form" are in fact Liouvillian.

An implementation of the Singer algorithm has not been achieved so far. Only the simpler *Kovačic algorithm* for second order equations has been implemented [69]. The main problem lies in the high complexity. The algorithm is based on the construction of a minimal polynomial for the logarithmic derivative of the solution. For this purpose first a bound on the degree of the polynomial is derived. Unfortunately this bound grows rapidly with the order of the equations.

Using some deep results from the representation theory of finite groups Ulmer [85] could improve the bounds given by Singer, so that at least the treatment of third order equations seems feasible. An alternative approach based on *invariants* was presented by Fakler [24]. For second order equations it leads in most cases to explicit solution formulae and thus to rather efficient algorithms. Some of them have meanwhile been implemented in *MuPAD*.[1]

The determination of the *differential Galois group* for a given equation is rather difficult. Some progress has been made for second and third order equations [81]. Solution algorithms like the one of Singer yield information about the group and in some cases actually determine it. If there was an easy way to compute the group directly, one could design more efficient algorithms.

In some sense related to the differential Galois theory is the problem of (efficiently) *factoring* linear differential operators [86]. All the theory mentioned here works only for irreducible equations. Thus before one can apply it, the equation must be factorized. The Newton polygon (see above) is here again quite useful. Factorization (although only of polynomials) is also an issue in differential ideal theory.

Numerical Analysis

It was already mentioned above that the capabilities of computer algebra systems to explicitly solve differential equations are limited. This holds especially for partial differential equations. Therefore numerical methods have lost nothing of their importance. Symbolic and numerical computations can interact in many ways and most computer algebra systems provide some numerical facilities, also for differential equations.

[1] Together with some related packages for linear differential operators they can be obtained from Fakler's WWW page with the URL `http://iaks-www.ira.uka.de/home/fakler/index.html`.

The oldest and simplest approach consists of *interfacing* a computer algebra system and a numerical library. Typically the interaction is one-way: the computer algebra system is used to derive the differential equations (e. g. the equations of motion of a complicated physical system); the interface generates code in the language of the numerical library (perhaps including some optimization steps); finally, the differential equations are solved with some methods from the numerical library.

To some extend this can be done with most of the common computer algebra systems, as they all provide commands to convert an expression into C or FORTRAN. However, it is rather cumbersome to automatically generate whole procedures or programs that way. The REDUCE package GENTRAN [30] can be used for such purposes. Another problem is the optimization of the generated code which is usually necessary. But again there exist already packages like SCOPE [87] for such tasks.

Computer algebra systems may further help to *select* an appropriate method from the library. Modern numerical libraries have reached such a level of sophistication that for many users it is increasingly difficult to fully exploit their potential. These libraries provide different routines for the same task and the working of these routines can be further tuned by many input parameters whose meaning remains a secret for non-experts. A computer algebra system can try to analyze the given differential equation (e. g. estimate its stiffness) and then choose an appropriate method and determine reasonable values for its parameters. An example for this approach is the AXIOM package ANNA developed by Dupée [22].

Goldman *et al.* [32] go considerably further in their application of computer algebra in numerical analysis by using it as a software engineering tool. They automatically *generate* the full code for numerically solving the Navier-Stokes equations. Their argument is that such programs are so long and complicated that their maintainance and adaption (new boundary conditions, different discretizations etc) is rather difficult and error-prone. They use instead a number of input files that contain all the relevant information about the problem in a format that is comparatively easy to read and let the computer algebra system then generate the source code.

Another approach consists of the use of computer algebra to *derive* new numerical schemes. The Butcher theory of Runge-Kutta methods provides here a typical example. For higher order methods the order conditions become rather large and complicated. Computer algebra packages are used to derive and solve them (using Gröbner bases) [82]. In the case of partial differential equations the construction of higher-order discretizations or finite elements can also be rather involved and is sometimes only feasible with the help of a computer algebra system [55]. Another application of computer algebra concerns the proof of the stability of a newly constructed scheme [29].

Another topic where computer algebra plays a certain role in numerical analysis are *differential algebraic equations*. The *index* of such a system comprising differential and algebraic equations measures in a certain sense, how far it is away from a pure differential equation [11]. This gives an indication of the difficulties one must expect in a numerical integration. The determination of the index is essentially equivalent to the completion procedures described above [47, 60], as it can be defined as the number of steps needed for the completion. However, in practice numerical analysts often prefer the use of *automatic differentiation* to computer algebra [12].

Somewhere in between numerical and symbolic methods are *approximation techniques*. Traditionally they are applied in a purely numerical fashion: some ansatz is made and then its coefficients are determined numerically. The use of computer algebra not only allows for much more complicated ansätze, but one can often computed the coefficients symbolically. This is especially valuable for parameter dependent problems. A numerical computation can be done only for fixed values of the parameters. A symbolic computation allows us to analyze the effect of changes in the parameters.

A Computer Algebra Environment for Differential Equations

Most computer algebra applications to differential equations published so far consist of a special purpose package devoted to one specific algorithm. Communication between different packages, e. g. further processing of the output of one algorithm by another one, is often rather awkward, as every package uses its own data structures and thus complicated conversion procedures must be implemented.

This problem could be avoided by providing an *environment* for computations with differential equations. Such an environment should comprise basic data structures and procedures for the representation of derivatives and differential equations. Ideally the user should have a choice, as different representations or notations are optimal in different problems.

Within this environment it would then be possible to implement packages for different purposes like completion, construction of the symmetry algebra, Painlevé analysis etc. Since all of them would be based on the same underlying data structures, they could easily communicate which each others. Furthermore this would considerably facilitate the implementation of further algorithms, as a large part of the work to develop, say, a new symmetry package consists just of writing these basic structures and procedures.

Such an environment can be reasonably developed only in an object-oriented system allowing for the simple implementation of abstract data types. In the language of computer algebra one often speaks of *domains* and *categories*. The latter ones are especially important, as they offer the possibility of generic programming. In *MuPAD* they are provided by the `domains` library [21].

We started with the development of an environment for differential equations, called JET, within Axiom [71, 74]. Currently it is ported to *MuPAD*.[2] As JET was originally designed for geometric approaches based on the jet bundle formalism, we will use a geometric language for its description. But JET is equally well suited for differential algebra, as the basic representation problems are exactly the same.

JET consists essentially of a three level hierarchy.[3] The properties of the two lower levels are each defined by a category; the third level contains the application packages. Besides there exist some utility domains, e. g. for sparse matrices, vector fields and differential forms, or for coordinate transformations in differential equations.

The lowest level is in some sense of a purely "cosmetic" nature. It defines only the notation used for jet variables (or differential unknowns) and provides procedures for their in- and output. It also introduces a ranking. One could argue whether this level is really necessary, but it allows for much more comfortable user interaction. Each user can implement a domain with his favorite notation and still make use of the full environment.

The second level concerns functions depending on the jet variables. Many operations like total derivatives or Jacobians are implemented categorically, i. e. if one implements a new domain for functions, there is no need to write routines for these tasks, as they are inherited from the category. But if it is possible to design a more efficient algorithm for some special class of functions, one may override the default implementation.

At this level one can see best the advantages of such an object-oriented approach. In many applications one encounters the same subproblems like for example the simplification of a system. Although the main algorithm (e. g. a completion procedure) runs the same way for any class of differential equations, these subproblems may be solved in very different ways: for linear systems with Gaussian elimination, for polynomial systems with Gröbner bases, for general systems with some sort of heuristics.

[2] Those parts of it that are already ported can be obtained from our *MuPAD* archive on the WWW under the URL `http://iaks-www.ira.uka.de/iaks-calmet/werner/mupad.html`.

[3] For efficiency reasons one sometimes includes a fourth level, but this will be ignored here.

In a classical programming language any special class of functions would need its own completion procedure. In an object-oriented language where domains can be passed as parameters to other domains the main algorithm is implemented only once. It takes a domain of functions as argument and uses for the subproblems the procedures provided by this domain.

The third level of JET contains applications. Currently it consists only of a package for a completion procedure based on the Cartan-Kuranishi Theorem and a symmetry package of classical and non-classical Lie point symmetries. Especially the latter one demonstrates nicely how fast packages can be developed within such an environment. The basic procedure setting up the determining system consists of about 15 lines of code! Admittingly it is not a very sophisticated procedure, but everything else needed in a symmetry package was already part of the environment.

Conclusions and Outlook

The application of computer algebra to differential equations is a vast field. We could only briefly indicate some of the main research directions. An important topic ignored here are e. g. *first integrals* [64, 51]. The idea of transforming differential equations can be extended far beyond simple heuristics leading to the *equivalence problem of Cartan* [45, 58]. There exist also much more applications of series methods than we could cover here.

The fields we have touched on are in rather different states. Some of them like symmetry theory are meanwhile fairly mature with the fundamentals well understood and they provide standard techniques for tackling differential equations implemented in many computer algebra systems. Others are still in an early stage of their development and essential questions are open. Such fields are usually known only to some experts and only prototypical implementations of algorithms exist.

One common feature shared by most of the fields mentioned is the complexity of the algorithms. If we take the various completion methods mentioned above as example, then it is obvious from their close relation to Gröbner bases that their complexity is at least as bad as that of the Buchberger algorithm. Although Gröbner bases solve *in principle* many problems in commutative algebra, it is well-known that one often fails to get a basis in reasonable time. One possible way out is the stronger use of heuristics and techniques from Artificial Intelligence, although this is an unpleasant thought for many pure mathematicians.

Some readers might be surprised that we discussed applications in numerical analysis as broadly as more traditional topics like symmetry theory. But we believe that in the future this direction will be among the most important ones. Despite all the successes of Lie symmetries, differential Galois theory etc. one must clearly see that these theories are of hardly any value for real world problems like for example the ones an engineer typically face.

A popular benchmark for the numerical integration of differential algebraic equations comes from vehicle dynamics and models with five links a wheel suspension [40]. Its equations of motion must be generated by computer and consist of 7000 lines of FORTRAN code. It seems hardly realistic to solve such a system with Lie symmetries (if it possesses any!) or any other analytic technique.

This does not imply that there is no point in further studying symbolic methods, not at all! Toy models that can be solved analytically are important for obtaining a deeper understanding of underlying structures. One may hope that such understanding may lead to more efficient numerical algorithms for the real world problems.

Most of the current numerical methods for ordinary differential equations do not take any special properties of the equation into account (with the possible exception of its stiffness). It is a rather new trend in numerical analysis to try to identify such properties and to use them to design more efficient algorithms.

One prominent example of this trend are *symplectic integrators* [68] which are superior to most conventional method in the long term integration of Hamiltonian systems, as they preserve many qualitative features of such systems.

The combination of symbolic and numerical computation will play in the future a much bigger role than currently. In the form of simple interfaces it happens already now in many places. For most *users* of computer algebra systems (this is a very different community than the participants of computer algebra conferences!) such possibilities are of much greater importance than many of the fancy algorithms developed by theorists.

Acknowledgments

I thank F. Postel for inviting me to write this article. W. Fakler, E. Hubert and E. Pflügel read parts of a draft version and gave valuable comments. This work was supported by Deutsche Forschungsgemeinschaft.

References

[1] M.J.D Ablowitz, A. Ramani, and H. Segur. Nonlinear evolution equations and ordinary differential equations of Painlevé type. *Lett. Nuovo Cim.*, 23:333–338, 1978.

[2] R.L. Anderson and N.H. Ibragimov. *Lie-Bäcklund Transformations in Applications.* SIAM, Philadelphia, 1979.

[3] A. Barkatou. A rational version of Moser's algorithm. In A.H.M. Levelt, editor, *Proc. ISSAC '95*, pages 297–302. ACM Press, New York 1995.

[4] M.A. Barkatou. An algorithm to compute the exponential part of a formal fundamental matrix solution of a linear differential system. *Appl. Alg. Eng. Comm. Comp.*, 8:1–23, 1997.

[5] O. Becken. Algorithmen zum Lösen einfacher Differentialgleichungen. *Rostocker Informatik Berichte*, 17:1–24, 1995.

[6] G.W. Bluman and J.D. Cole. The general similarity solution of the heat equation. *J. Math. Mech.*, 18:1025–1042, 1969.

[7] G.W. Bluman and S. Kumei. *Symmetries and Differential Equations.* Applied Mathematical Sciences 81. Springer-Verlag, New York, 1989.

[8] F. Boulier, D. Lazard, F. Ollivier, and M. Petitot. Representation for the radical of a finitely generated differential ideal. In A.H.M. Levelt, editor, *Proc. ISSAC '95*, pages 158–166. ACM Press, New York 1995.

[9] M. Bronstein. Linear ordinary differential equations: breaking through the order 2 barrier. In B.M. Trager and D. Lazard, editors, *Proc. ISSAC '92*, pages 42–48. ACM, New York 1992.

[10] R.L. Bryant, S.S. Chern, R.B. Gardner, H.L. Goldschmidt, and P.A. Griffiths. *Exterior Differential Systems.* Mathematical Sciences Research Institute Publications 18. Springer-Verlag, New York, 1991.

[11] S.L. Campbell and C.W. Gear. The index of general nonlinear DAEs. *Numer. Math.*, 72:173–196, 1995.

[12] S.L. Campbell and R. Hollenbeck. Automatic differentiation and implicit differential equations. In *Proc. Second Int. Workshop Computational Differentiation, Santa Fe 1996*, to appear.

[13] J. Carminati, J.S. Devitt, and G.J. Fee. Isogroups of differential equations using algebraic computing. *J. Symb. Comp.*, 14:103–120, 1992.

[14] G. Carrà Ferro. Gröbner bases and differential algebra. In L. Huguet and A. Poli, editors, *Proc. AAECC-5*, Lecture Notes in Computer Science 350, pages 129–140, Berlin, 1987. Springer-Verlag.

[15] P.A. Clarkson and M.D. Kruskal. New similarity reductions of the Boussinesq equation. *J. Math. Phys.*, 30:2201–2212, 1989.

[16] R. Conte. Singularities of differential equations and integrabilty. In D. Benest and C. Froeschlé, editors, *Introduction to Methods of Complex Analysis and Geometry for Classical Mechanics and Non-Linear Waves*, pages 49–143. Editions Frontières, Gif-sur-Yvette, 1994.

[17] R. Conte, A.P. Fordy, and A. Pickering. A perturbative Painlevé approach to nonlinear differential equations. *Physica D*, 69:33–58, 1993.

[18] J.H. Davenport, Y. Siret, and E. Tournier. *Computer Algebra*. Academic Press, London, 1988.

[19] J. Denef and L. Lipshitz. Power series solutions of algebraic differential equations. *Math. Ann.*, 267:213–238, 1984.

[20] J. Della Dora and E. Tournier. Formal solutions of differential equations in the neighbourhood of singular points. In P.S. Wang, editor, *Proc. SYMSAC*, pages 25–29. ACM, New York 1981.

[21] K. Drescher. Axioms, categories and domains. Automath Technical Report No. 1, Universität Paderborn, 1996.

[22] B.J. Dupée and J.H. Davenport. An intelligent interface to numerical routines. In J. Calmet and C. Limongelli, editors, *Proc. DISCO '96*, Lecture Notes in Computer Science 1128. Springer, Heidelberg.

[23] P.G. Estévez. Non-classical symmetries and the singular manifold method: the Burgers and the Burgers-Huxley equations. *J. Phys. A*, 27:2113–2127, 1994.

[24] W. Fakler. On second order homogeneous linear ordinary differential equations with Liouvillian solutions. *Theor. Comp. Sci.*, to appear.

[25] A.S. Fokas. Symmetries and integrability. *Stud. Appl. Math.*, 77:253–299, 1987.

[26] B. Fuchssteiner, S. Ivanov, and W. Wiwianka. Algorithmic determination of infinite dimensional symmetry groups for integrable systems in 1+1 dimensions. *Math. Comp. Model.*, to appear, 1997.

[27] A. Galligo. Some algorithmic questions on ideals of differential operators. In B.F. Caviness, editor, *Proc. EUROCAL '85, vol. 2*, Lecture Notes in Computer Science 204, pages 413–421. Springer-Verlag, Berlin 1985.

[28] G. Gallo, B. Mishra, and F. Ollivier. Some constructions in rings of differential polynomials. In H.F. Mattson, T. Mora, and T.R.N. Rao, editors, *Proc. AAECC-9*, Lecture Notes in Computer Science 539, pages 171–182. Springer-Verlag, Heidelberg 1991.

[29] V.G. Ganzha and R. Liska. Application of the Reduce computer algebra system to stability analysis of difference schemes. In E. Kaltofen and S.M. Watt, editors, *Computers and Mathematics*, pages 119–129. Springer, New York, 1989.

[30] B.L. Gates. A numerial code generation facility for REDUCE. In B.W. Char, editor, *Proc. SYMSAC '86*, pages 94–99. ACM Press, New York 1986.

[31] V.P. Gerdt. Gröbner bases and involutive methods for algebraic and differential equations. In J. Fleischer, J. Grabmeier, F.W. Hehl, and W. Küchlin, editors, *Computer Algebra in Science and Engineering*, pages 117–137. World Scientific, Singapore, 1995.

[32] V.V. Goldman, J.A. van Hulzen, A.E. Mynett, A.S. Posthuma, and H.J. van Zuylen. The application of computer algebra for the discretization and coding of the Navier-Stokes equations. In A.M. Cohen, L. van Gastel, and S. Verduyn Lionel, editors, *Computer Algebra in Industry 2*, pages 131–150. John Wiley, New York, 1995.

[33] B.K. Harrison and F.B. Estabrook. Geometric approach to invariance groups and solutions of partial differential systems. *J. Math. Phys.*, 12:653–666, 1971.

[34] D.H. Hartley. EDS: A REDUCE package for exterior differential systems. *Comp. Phys. Comm.*, 100:177, 1997.

[35] D.H. Hartley and R.W. Tucker. A constructive implementation of the Cartan-Kähler theory of exterior differential systems. *J. Symb. Comp.*, 12:655–667, 1991.

[36] A.K. Head. Lie, a PC program for Lie analysis of differential equations. *Comp. Phys. Comm.*, 77:241–248, 1993.

[37] W. Hereman. Review of symbolic software for the computation of Lie symmetries of differential equations. *Euromath Bull.*, 2:45–82, 1994.

[38] W. Hereman. Symbolic software for Lie symmetry analysis. In N.H. Ibragimov, editor, *CRC Handbook of Lie Group Analysis of Differential Equations, Volume 3: New Trends in Theoretical Development and Computational Methods*, chapter 13. CRC Press, Boca Raton, Florida, 1995.

[39] W. Hereman and E. Van den Bulck. MACSYMA program for the Painlevé test of nonlinear ordinary and partial differential equations. In P.G.L. Leach and W.H. Steeb, editors, *Proc. Finite Dimensional Integrable Nonlinear Dynamical Systems*, pages 117–129, Johannesburg, South Africa, 1988. World Scientific, Singapore 1988.

[40] M. Hiller and S. Frik. Road vehicle benchmark 2: Five link suspension. In W. Kortüm, S. Sharp, and A. de Pater, editors, *Application of Multibody Computer Codes to Vehicle System Dynamics, Progress Report to the 12th IAVSD Symposium, Lyon 1991*, pages 198–203, 1992.

[41] E. Hubert. The general solution of an ordinary differential equation. In Y.N. Lakshman, editor, *Proc. ISSAC '96*, pages 189–195. ACM, New York 1996.

[42] E.L. Ince. *Ordinary Differential Equations*. Dover, New York, 1956.

[43] M. Janet. Sur les Systèmes d'Équations aux Dérivées Partielles. *J. Math. Pure Appl.*, 3:65–151, 1920.

[44] E. Kamke. *Differentialgleichungen: Lösungsmethoden und Lösungen*, volume 1. Geest & Portig, Leipzig, 1967.

[45] N. Kamran. The equivalence problem of Elie Cartan, differential equations and computer algebra. In E. Tournier, editor, *Computer Algebra and Differential Equations*, pages 87–114. Academic Press, London, 1988.

[46] E.R. Kolchin. *Differential Algebra and Algebraic Groups*. Academic Press, New York, 1973.

[47] G. Le Vey. Differential algebraic equations: A new look at the index. Rapport de Recherche 2239, INRIA Rennes, 1994.

[48] H. Lewy. An example of a smooth linear partial differential equation without solution. *Ann. Math.*, 66:155–158, 1957.

[49] I.G. Lisle, G.J. Reid, and A. Boulton. Algorithmic determination of structure of infinite Lie pseudogroups of symmetries of PDEs. In A.H.M. Levelt, editor, *Proc. ISSAC '95*, pages 1–6. ACM press 1995.

[50] M.A.H. MacCallum. Using computer algebra to solve ordinary differential equations. In A.M. Cohen, L. van Gastel, and S.M. Verduyn Lunel, editors, *Computer Algebra in Industry 2*, pages 19–41. John Wiley, New York, 1995.

[51] Y.-K. Man. Computing closed form solutions of first order ODEs using the Prelle-Singer procedure. *J. Symb. Comp.*, 16:423–443, 1993.

[52] E. Mansfield. *Differential Gröbner Bases*. PhD thesis, Macquarie University, Sydney, 1991.

[53] J.B. McLeod and P.J. Olver. The connection between partial differential equations soluble by inverse scattering and ordinary differential equations of Painlevé type. *SIAM J. Math. Anal.*, 14:488–506, 1983.

[54] A.V. Mikhailov, A.B. Shabat, and R.I. Yamilov. The symmetry approach to the classification of non-linear equations. Complete lists of integrable systems. *Russ. Math. Surv.*, 42:3–53, 1987.

[55] E.H. Mund. Computer algebra and finite element methods in engineering. In A.M. Cohen, L. van Gastel, and S. Verduyn Lionel, editors, *Computer Algebra in Industry 2*, pages 81–100. John Wiley, New York, 1995.

[56] P.J. Olver. *Applications of Lie Groups to Differential Equations*. Graduate Texts in Mathematics 107. Springer-Verlag, New York, 1986.

[57] P.J. Olver. Direct reduction and differential constraints. *Proc. Roy. Soc. Ser. A*, 444:509–523, 1994.

[58] P.J. Olver. *Equivalence, Invariants, and Symmetry*. Cambridge University Press, 1995.

[59] E. Pflügel. On the latest version of DESIR. *Theor. Comp. Sci.*, to appear.

[60] O.-P. Piirilä and J. Tuomela. Differential-algebraic systems and formal integrability. Research Report A326, Helsinki University of Technology, Institute of Mathematics, 1993.

[61] J.F. Pommaret. *Systems of Partial Differential Equations and Lie Pseudogroups*. Gordon & Breach, London, 1978.

[62] J.F. Pommaret. *Differential Galois Theory*. Gordon & Breach, London, 1983.

[63] F. Postel and P. Zimmermann. A review of the ODE solvers of Maple, Mathematica, Macsyma and MuPAD. In A. Carriére and L.R. Oudin, editors, *Proc. 5th Rhine Workshop on Computer Algebra*, pages 2.1–2.10. Report PR 801/96, ISL, Saint-Louis (France), 1996.

[64] M.J. Prelle and M.F. Singer. Elementary first integrals of differential equations. *Trans. Amer. Math. Soc.*, 139:167–189, 1969.

[65] G.J. Reid. Algorithms for reducing a system of PDEs to standard form, determining the dimension of its solution space and calculating its Taylor series solution. *Eur. J. Appl. Math.*, 2:293–318, 1991.

[66] G.J. Reid, I.G. Lisle, A. Boulton, and A.D. Wittkopf. Algorithmic determination of commutation relations for Lie symmetry algebras of PDEs. In B.M. Trager and D. Lazard, editors, *Proc. ISSAC '92*, pages 63–68. ACM, New York 1992.

[67] J.F. Ritt. *Differential Algebra.* Dover, New York, 1966. (Original: AMS Colloquium Publications, Vol. XXXIII, 1950).

[68] J.M. Sanz-Serna and M.P. Calvo. *Numerical Hamiltonian Problems.* Applied Mathematics and Mathematical Computation 7. Chapman&Hall, London, 1994.

[69] B.D. Saunders. An implementation of Kovačic's algorithm for solving second order linear homogeneous differential equations. In P.S. Wang, editor, *Proc. SYMSAC*, pages 105–108. ACM, New York 1981.

[70] C. Scheen. Implementation of the Painlevé test for ordinary differential equations. *Theor. Comp. Sci.*, to appear.

[71] J. Schü, W.M. Seiler, and J. Calmet. Algorithmic methods for Lie pseudogroups. In N. Ibragimov, M. Torrisi, and A. Valenti, editors, *Proc. Modern Group Analysis: Advanced Analytical and Computational Methods in Mathematical Physics*, pages 337–344, Acireale (Italy), 1992. Kluwer, Dordrecht 1993.

[72] F. Schwarz. Automatically determining symmetries of partial differential equations. *Comp.*, 34:91–106, 1985.

[73] W.M. Seiler. On the arbitrariness of the general solution of an involutive partial differential equation. *J. Math. Phys.*, 35:486–498, 1994.

[74] W.M. Seiler. Applying AXIOM to partial differential equations. Internal Report 95-17, Universität Karlsruhe, Fakultät für Informatik, 1995.

[75] W.M. Seiler. Arbitrariness of the general solution and symmetries. *Acta Appl. Math.*, 41:311–322, 1995.

[76] W.M. Seiler. Involution and symmetry reductions. *Math. Comp. Model.*, to appear, 1997.

[77] M.F. Singer. Liouvillian solutions of n-th order homogeneous linear differential equations. *Am. J. Math.*, 103:661–682, 1981.

[78] M.F. Singer. Formal solutions of differential equations. *J. Symb. Comp.*, 10:59–94, 1990.

[79] M.F. Singer. An outline of differential Galois theory. In E. Tournier, editor, *Computer Algebra and Differential Equations.* Academic Press, New York, 1990.

[80] M.F. Singer. Liouvillian solutions of linear differential equations with Liouvillian coefficients. *J. Symb. Comp.*, 11:251–273, 1991.

[81] M.F. Singer and F. Ulmer. Galois groups of second and third order linear differential equations. *J. Symb. Comp.*, 16:9–36, 1993.

[82] M. Sofroniou. Symbolic derivation of Runge-Kutta methods. *J. Symb. Comp.*, pages 265–296, 1994.

[83] H. Stephani. *Differential Equations: Their Solution Using Symmetries.* Cambridge University Press, Cambridge, 1989.

[84] V.L. Topunov. Reducing systems of linear differential equations to a passive form. *Acta Appl. Math.*, 16:191–206, 1989.

[85] F. Ulmer. On Liouvillian solutions of linear differential equations. *Appl. Alg. Eng. Comm. Comp.*, 2:171–194, 1992.

[86] M. van Hoeij. *Factorization of Linear Differential Operators.* PhD thesis, Department of Mathematics, University of Nijmegen, 1996.

[87] J.A. van Hulzen, B.J.A. Hulshof, B.L. Gates, and M.C. van Heerwaarden. A code optimization package for REDUCE. In G.H. Gonnet, editor, *Proc. ISSAC '89*, pages 163–170. ACM Press, New York 1989.

[88] E. von Weber. Partielle Differentialgleichungen. In *Enzyklopädie der mathematischen Wissenschaften, Vol. II, Part 1.1*, chapter A5, pages 294–399. 1900.

[89] J. Weiss. The Painlevé property for partial differential equations II: Bäcklund transformations, Lax pairs, and the Schwarzian derivative. *J. Math. Phys.*, 24:1405–1413, 1983.

[90] J. Weiss, M. Tabor, and G. Carnevale. The Painlevé property for partial differential equations. *J. Math. Phys.*, 24:522–526, 1983.

[91] W. Wiwianka and B. Fuchssteiner. Algorithms to detect complete integrability in 1+1 dimensions. In S. Carillo and O. Ragnisco, editors, *Research Reports in Physics – Nonlinear Dynamics*, pages 131–135. Springer-Verlag, Berlin, 1990.

[92] Wu W.-T. Automatic derivation of Newton's gravitational law from Kepler's laws. *Academica Sinica Mathematics-Mechanization Research Preprints*, 1:53, 1987.

[93] Wu W.-T. A constructive theory of differential algebraic geometry based on works of J.F. Ritt with particular applications to mechanical theorem proving of differential geometries. In G. Chaohao, M. Berger, and R.L. Bryant, editors, *Differential Geometry and Differential Equations*, Lecture Notes in Mathematics 1255, pages 173–189, Shanghai, 1987. Springer-Verlag, Berlin 1987.

[94] D. Zwillinger. *Handbook of Differential Equations*. Academic Press, San Diego, 1992.