Computations and Some Open Problems

William Sit, City College of New York

April 28, 2006

Lecture as part of Graduate Center Series
For Kolchin Seminar in Differential Algebra, 2005–6

Abstract

I'll discuss some open problems related to differential algebra for which experiments in computations may be helpful. Examples showing how to set up such experiments will be given.

There are two obvious points I want to make:

1. Computations become necessary when we want to compute something, like solving a system of (differential) equations, or finding some examples or counterexamples.
2. In turns, computational consideration of effectiveness and efficiency may itself lead to mathematical problems, such as complexity, enumeration of vector space basis, encoding of input and output, etc. Some of these problems are of interest in its own right.

Advice: You can write a dissertation or paper on the new problems if you cannot solve the original computation problem!

Gaussian Elimination

Let's illustrate the two points with linear systems. Linear algebra hides behind almost all symbolic computations: for example, partial fraction decomposition (used in integration of rational functions and inverse Laplace transforms) and the method of undetermined coefficients (used in solving non-homogeneous linear differential equations with constant coefficients).

As is well known, we can solve a linear system by Gaussian elimination or by Cramer's rule. Both algorithms are straight forward (indeed, the symbolic computation versions are much simpler than the numerical version since one does not have to worry about round off errors). **The complexity[1] of the Gaussian elimination algorithm is $\mathcal{O}(n^3)$ where n is the number of equations and unknowns, and the complexity is measured by the number of arithmetic operations like multiplication.** Depending on the coefficient domain over which arithmetic is performed, such operations may be simple, or may involve rather complicated algorithms.

The complexity of Cramer's Rule, when determinants are computed by definition, is $n + (n-1)(n+1)!$ multiplications ($n-1$ multiplications for each of the $n!$ generalized diagonals for each of $n+1$ determinants). With other methods, **the complexity to compute the determinant is roughly $\mathcal{O}(n^3)$**, see Kaltofen and Villard [2, 3].

In any case, **Cramer's Rule is a no-no for computations.**

[1] For detail counts of multiplications and additions due to variations of the method, see Tapia and Lanius [4].

Branching in Gaussian Elimination

A natural question that arises is to **determine the number of distinct ways the Gaussian elimination algorithm may be executed when applied to all possible linear systems of a given size**. Formally, let

$$x_{i1}z_1 + \cdots + x_{in}z_n = w_i \qquad (1 \leq i \leq r) \tag{1}$$

be a linear system of r equations in n indeterminates $z_1, \ldots, z_n$. Of the three types of elementary row transformations used in any Gaussian elimination scheme, we have to be particularly careful with the one that **multiplies (or divides)** a row by a "non-zero" entry, which will be some polynomial function g of the coefficients x_{ij}. In a computer algorithm, each such transformation must be considered a **pivoting step** and the process must branch based on a test of whether g evaluates to zero or not (unless g is a constant independent of x_{ij}). Let G be the binary tree corresponding to the Gaussian elimination algorithm: each non-zero, non-constant pivot element g is represented by a node, with a left branch (if g evaluates to non-zero) and a right branch (if g evaluates to zero); a branch either leads to a node which is the next pivot in sequence or to a leaf which represents a decision that the system is consistent or not consistent. **The number of leaves in G is the number of all possible ways the Gaussian elimination algorithm may be executed when applied to *all* possible linear systems of the given size.**

Parametric Linear System

Another way to interpret the number of leaves of G is to consider

$$x_{i1}z_1 + \cdots + x_{in}z_n = w_i \qquad (1 \leq i \leq r) \tag{1}$$

as a **generic parametric linear system**, with parameters (x_{ij}) and w_i. In a parametric system, **solving the system means determining conditions on the parameters so that the system is consistent.** Instead of evaluating g at each non-constant pivot, we branch by **specifying** that the parameters either satisfy $g \neq 0$ or $g = 0$. For each path, we collect the conditions specified along the path. If these can be satisfied (which requires testing whether 1 belongs to the polynomial ideal related to the g's), then the leaf at the end of this path represents a consistent system when the parameters are suitably specialized. The number of leaves then measures the number of different paths that must be walked through to completely determine all **the conditions for consistency**. The number of leaves for the generic system (1) is a measure of the **worst case complexity** of the Gaussian elimination algorithm.

Notice for each path, determining consistency is a **non-linear problem**, requiring Gröbner basis computations in a large number of indeterminates (parameters). In other words, it is very **expensive**!

So, **what is this worst case complexity?**

Worst Case Complexity

For the convenience of the derivation, we shall measure the size of (1) by the size of the augmented matrix. If the **augmented matrix has size** $r \times n$ (note change in notation), let $\varphi(r, n)$ **be the number of leaves**. We know easily $\varphi(1, n) = n + 1$ and we define $\varphi(r, 1)$ to be $r + 1$, which is the number of paths of the Gaussian algorithm when the coefficient matrix is the zero matrix and the right hand side is a vector of indeterminates (r of these paths will lead to inconsistency, and the remaining path leads to the unique trivial solution). Using an induction argument, we can show that

$$\varphi(r, n) = r\varphi(r - 1, n - 1) + \varphi(r, n - 1) \qquad (r \geq 2, n \geq 2) \tag{2}$$

This is a **partial difference equation with variable coefficients**. In elementary combinatorics courses, we learnt how to solve such a recursion using generating functions. It is my experience, however, that it is usually the case that the methods can only be applied to the text book cases and never to the one that you want to solve. I leave this as an exercise, but I'll tell you that the answer is

$$\varphi(r, n) = \sum_{i=0} \binom{r}{i} \binom{n}{i} i!. \tag{3}$$

This formula is difficult to find, but can be proved very easily using induction. Applying this to (1) (where n now means the number of unknowns again), the number of leaves is

$$\varphi(r, n + 1) = \sum_{i=0} \binom{r}{i} \binom{n}{i} i!(r - i + 1). \tag{4}$$

Some Open Problems

$$\varphi(r,n) = \sum_{i=0} \binom{r}{i}\binom{n}{i} i!. \tag{3}$$

$$\varphi(r,n+1) = \sum_{i=0} \binom{r}{i}\binom{n}{i} i!(r-i+1). \tag{4}$$

Here's some questions that come to mind:

1. We have $\varphi(r,n) = \varphi(n,r)$. Why is the function symmetric in r,n? Is this obvious in some way?
2. Can the formula for $\varphi(r,n+1)$ be given in some closed form?
3. Is there a combinatorial way to interpret the result in (3) and (4)?

Evaluating Subdeterminants

A very naive method to determine the consistency conditions of a generic parametric linear system is the following: for each i, $0 \leq i \leq \min(r, n)$, *specify* that the rank of the system be i. Then every $(i+1) \times (i+1)$ subdeterminant must be zero (this condition is vacuous if $i = \min(r, n)$) and there is at least one $i \times i$ subdeterminant that is non-zero. Now there are precisely

$$\sum_{i=0}^{\min(r,n)} \binom{r}{i}\binom{n}{i} = \binom{n+r}{r} \tag{5}$$

subdeterminants. Each one leads to one set of consistency conditions and the totally covers all possibilities. If we compare the number of cases with the Gaussian elimination method, we see that this naive method reports a lot less cases.

$n = r$	$\binom{n+r}{r}$	$\sum \binom{n}{i}\binom{r}{i} i!(r-i+1)$
1	2	3
2	6	13
3	20	73
4	70	501
5	252	4,051
6	924	37,633
7	3,432	394,353
8	12,870	4,596,553
9	48,620	58,941,091
10	184,756	824,073,141

Axiom Package: PLEQN

The package PLEQN in the computer algebra system *Axiom* [1] is a proof-of-concept implementation of the naive algorithm. *Axiom* allows the same implementation to work across multiple choices of data structure representing mathematical objects such as polynomials.

In the demo, we solve the following parametric system, using different term orderings for Gröbner bases computations to determine consistency at the leaves and also different polynomial representations. It also solves the equations when the right hand side is replaced by arbitrary unknowns.

$$\begin{aligned} z_1 + az_2 + bz_3 &= 1 \\ bz_1 + z_2 + az_3 &= 1 \\ az_1 + bz_2 + z_3 &= 1 \end{aligned}$$

The system computes the set of equilibrium points of a Lotka-Volterra system studied in Gardini *et al.* [5]. The computer program did better than the analysis in that paper (see Sit [6]).

The output gives the complete conditions on a, b (and w_1, w_2, w_3) under which the system is consistent, and the corresponding solutions when it is consistent.

First Integrals of Dynamical Systems

Laws of conservation are important results in both mechanics and physics. Many of these are first integrals of a system of differential equations. A well-known class is the class of Hamiltonian systems. The system of differential equations can usually be transformed into a multinomial autonomous form:

A **multinomial autonomous system** is a system of first order differential equations of the form

$$y_i' = y_i \sum_{k=1}^{r} c_{i,k} y_1^{h_{k,1}} \cdots y_n^{h_{k,n}}, \quad i = 1, \ldots, n \tag{6}$$

where $c_{i,k}$ and $h_{k,i}$ are constants (perhaps unknown) and $\mathbf{Y} = (y_1, \ldots, y_n)$ are n functions of an independent variable t. We shall assume that the there are no algebraic relations among the y_i over constants.

Let H_k be the row vector $(h_{k,1}, \ldots, h_{k,n})$. Let C_k be the column vector $(c_{1,k}, \ldots, c_{n,k})$. We write the system (6) by the short hand notation as

$$S: \qquad \mathbf{Y}' = \mathbf{Y} \sum_{k=1}^{r} C_k \mathbf{Y}^{H_k}. \tag{7}$$

A *first integral* **for** S is in general a function $I(y_1, \ldots, y_n)$ such that $dI/dt = 0$. In this talk, we only consider first integrals that can be expressed as a linear combination of monomials in $\mathbf{Y}$.

Schwarz's Algorithm

If we assume the first integrals are polynomial in $\mathbf{Y}$, then we can use the method of undetermined coefficients. This is the method of Schwarz. The problem with this method is that even for a very low total degree bound, the number of monomials in n variables of degree less than or equal to d is given by

$$\binom{n+d}{d}$$

and this easily gets too many monomials with the result being a very large parametric linear system, too large for any Gröbner basis computation. Schwarz devised an elaborate method to perform elimination for the resulting linear system. However, his method cannot handle dynamical systems with parameters.

But very often, even though the degree in a first integral may be large, there are usually very few terms. Moreover, they may involve rational exponents.

Goldman's Algorithm

Let a potential first integral be represented by:

$$I = \sum_{j=1}^{q} e_j \mathbf{Y}^{B_j}, \tag{8}$$

where B_j is the row vector $(b_{j,1}, \ldots, b_{j,n})$ and $b_{j,i}, e_j$ are constants to be determined for a fixed given q.

For any monomial $\mathbf{Y}^B$, it is trivial to show that

$$(\mathbf{Y}^B)' = \sum_{k=1}^{r} (B, C_k) \mathbf{Y}^{B+H_k} \tag{9}$$

and hence for the first integral I given by (8),

$$\begin{aligned} I' &= \sum_{j=1}^{q} \sum_{k=1}^{r} e_j (B_j, C_k) \mathbf{Y}^{B_j+H_k} \\ &= \sum_{h=1}^{s} \sum_{B_j+H_k=E_h} e_j (B_j, C_k) \mathbf{Y}^{E_h}, \end{aligned} \tag{10}$$

where in the last summation, $E_1, \ldots, E_s$ is a complete enumeration of the set

$$\{\, B_j + H_k \,|\, 1 \le j \le q, 1 \le k \le r \,\}.$$

Collecting Like Terms

How can we set up equations to find B_j and e_j? or the parameters in the given dynamical system?

Clearly, we can write down some equations for the coefficients *if we only knew how to collect the like terms in*

$$I' = \sum_{h=1}^{s} \sum_{B_j + H_k = E_h} e_j(B_j, C_k)\mathbf{Y}^{E_h} \tag{10}$$

But the exponents in the terms involve the unknown vectors B_j, not to mention that the exponents H_k of the original system may also contain parameters to be chosen. This **dilemma is solved by** *arbitrarily specifying* **a collection rule** in the form of what I called an addition scheme (L. Goldman, who discovered this in 1987, has a slightly different set up: integral array).

Clearly **the set $\mathcal{A}_{q,r}$ of all $q \times r$ addition schemes is finite**. Our interest in addition schemes is that **they give us a complete list of possible ways the like terms in (10) may collect**.

The Magic of Addition Schemes

:

It turns out that **a first integral determines, and up to a constant factor can be determined by the addition scheme** $[B, H]$, where $B = \{ B_1, \ldots, B_q \}$ and $H = \{ H_1, \ldots, H_r \}$. Given the addition scheme, we can find out **even the conditions on the parameters embedded in** C_k, H_k under which the dynamical system is guaranteed to have a first integral, using the algorithm to solve parametric linear systems. Moreover, **we can find these first integrals**.

All the algorithms only involve linear algebra!

My favorite example for Goldman's algorithm is getting all the known first integral cases of the Lorenz system, and finding a new one!

Open Problems in Addition Schemes

Abstract addition schemes can be defined as isomorphism classes of addition schemes over any all additive groups M. The **combinatorial problems** are:

1. **How do we represent addition schemes?**
2. **How do we enumerate all addition schemes of a given size?**
3. **How many are there?**
4. **Can every addition scheme be realized over $\mathbb{Z}$?**

I have not tried to solve these general problems. Since my application is to compute first integrals, and in computer algorithms, every list must be ordered in some way, I concentrate only on the special cases when M is an ordered abelian group. This restriction ignores all torsion groups. According to a **theorem of Hahn, every ordered (actually lattice-ordered) abelian group is isomorphic to a product of subgroups of the real numbers with lexicographic ordering.** So the problems may be easier for ordered abelian groups.

In the first integral setting, we **can represent addition schemes using integer matrices**, where the entries are h if $B_j + H_k = E_h$. It is not difficult to use a **backtracking method** to generate a list of matrices that *includes* all representations of addition schemes and to use linear algebra to remove those that are not.

The third and fourth questions are open. I believe the answer to the fourth is affirmative.

Goldman's Algorithm in Axiom

I have implemented a proof-of-concept version of the algorithm in *Axiom*. There are a few steps:

1. Given r, q, generate all $r \times q$ addition schemes.
2. Given an $r \times q$ addition scheme S, test it to see if it matches the addition scheme of the "given" B_j, H_k (we are allowed to include consistent conditions on the parameters and the addition scheme S provides them).
3. Repeat Step 2 for all possible $r \times q$ addition schemes. If there is a first integral of the form I, there has to be one.
4. Once a matching addition scheme S is found, solve the parametric linear system in B_j and then solve another linear system for e_j.

Open Problems:

- **Can we find the right addition schemes (the matching problem)** *without* **an exhaustive search?**
- **What is the complexity of the matching problem for dynamical system? in general? Is the problem NP or even NP complete?**

Lattice Points, Monomials, and Partial Derivatives

A prerequisite to any computations involving systems of algebraic equations, or the more complicated systems of differential algebraic equations is a method to order the terms. For polynomial rings $K[x_1, \ldots, x_m]$, this is called a **term-ordering**. A typical example is the pure lexicographical order in which we say a monomial $x_1^{a_1} \cdots x_m^{a_m} < x_1^{b_1} \cdots x_m^{b_m}$ if the m-tuple $(a_1, \ldots, a_m)$ is lexicographically less than $(b_1, \ldots, b_m)$. In this ordering, $x_1 > x_2 > \cdots > x_m$ and any order on the set of variables can induce a corresponding pure lexicographical term ordering. **There is a natural bijection between the set of monomials and the lattice points in N^m.**

A differential polynomial ring $K\{ y_1, \ldots, y_n \}$ is constructed from a polynomial ring $K[\mathcal{Y}]$ where $\mathcal{Y}$ is the set of all derivatives of $y_1, \ldots, y_n$. Thus it is a polynomial ring in infinitely many algebraic indeterminates. If the set of derivations is $\Delta = \{ \delta_1, \ldots, \delta_m \}$, then a partial derivative of y_k has the form $\delta_1^{a_1} \cdots \delta_m^{a_m} y_k$. The easiest way to give a term-ordering is to first give an order on the set $\mathcal{Y}$ of derivatives (this order is called a **ranking**) and then use the induced pure lexicographical ordering.

Despite the similarity between a partial derivative and an algebraic monomial, it is not easy to characterize all rankings. I will not go into this topic today. What I like to point out is that **there is a natural bijection between $\mathcal{Y}$ and the set of lattice points in n copies of N^m.**

Hilbert Polynomials

When we are given an **algebraic system of equations**, the set of solutions is called an algebraic set. As we all know, we cannot really solve the system in general. What we can do is to reexpresss the system in other forms through a process of algebraic elimination similar to Gaussian elimination. Once such process is the **Buchberger algorithm for Gröbner basis**.

An algebraic system of equations may be thought of as a linear system whose "variables" are the monomials. This linear system is made up of the given equations, and all equations that can be obtained from these by multiplying with one or more of the variables. Put another way, if we just begin with the original equations, then during Gaussian elimination of the monomials, we are allowed to multiply each row (equation) not only by a scalar in K but also by any monomial. **A Gröbner basis may be viewed roughly as the reduced row echelon form at the end of this process.**

The important concept here is that the **"variables with a leading 1"** are now leading monomials of a polynomial in the ideal generated by the algebraic system. The non-leading monomials are **"free."** The number $H(s)$ of such free monomials of degree $\leq s$ is an important measure of the size of the algebraic set and H is called the **Hilbert function** of the system. It can be shown that there is a finite number of leading monomials such that all other leading monomials are multiples of these (This is Hilbert's Basis Theorem) and that $H(s)$ **is a polynomial** H **in** s **when** s **is sufficiently large.**

Computing the Hilbert Polynomial

Now the combinatorial problem is to compute $H(s)$ or the Hilbert series $\sum H(s)z^s$ when the leading monomials are known. This means given a finite set of k lattice points in N^m, and a positive integer s, find the number of points of degree $\leq s$ which are not in the positive cones spanned by the lattice points.

One method is based on the **inclusion-exclusion formula**. Another one is based on recursion. But neither of the methods are polynomial in time. The first has complexity $\mathcal{O}(2^k)$ while the other has complexity $\mathcal{O}(k^m)$. It would be very desirable to find more efficient methods.

A priori knowing the Hilbert function can speed up the computation of Gröbner basis. Random homogeneous algebraic systems are likely to be complete intersections, and for these, the Hilbert function is known.

Thus, if we can compute a Gröbner basis in one term-ordering, use it to compute the Hilbert function, we can apply this to find the Gröbner basis in another term-ordering, possibly faster than a direct recomputation.

Differential Dimension Polynomials

In a similar way, when we are given a system of differential algebraic equations, the set of solutions is called a **differential algebraic set**. We can view the system as an algebraic system in the algebraic indeterminates $\mathcal{Y}$ and apply elimination similar, but different from, the Gröbner basis method. For differential systems, we need to include the derivatives of the given equations.

Fortunately, when we differentiate an equation, the result is always a differential equation that is linear in the leading derivative. **The aim of differential elimination is then to solve for these leading derivatives in terms of non-leading derivatives.** Using the characteristic set method, we can **determine a finite set of leading derivatives so that all other leading derivatives are derivatives of these**.

The number $\omega(s)$ of non-leading derivatives that have order less than or equal to s is an important measure of the size of the differential algebraic set. In the case that there are only a finite number of non-leading derivatives, these derivatives may be assigned arbitrary values (initial conditions) and the rest of the derivatives can then be computed in terms of them. It is the basis of numerical methods. In the more general setting, $\omega(s)$ grows as a polynomial in s when s is sufficiently large and is known as the differential dimension polynomial of the system. If for some dependent variable y_i, all its derivatives are non-leading, we can choose y_i to be an arbitrary function of m independent variables.

The function $\omega(s)$ is a numerical polynomial when s is sufficiently large and the polynomial is called the **differential dimension polynomial** or **Kolchin polynomial** of the system.

Example and Open Problem

As an example, suppose we have in $\mathbb{N}^2$ the lattice points $(4,4),(7,2),(8,0)$. These lattice points may be thought of as the basis of leading monomials x^4y^4, x^7y^2, x^8 from a system of algebraic equations in x, y, or as the basis of leading derivatives

$$\partial_x{}^4\partial_y{}^4 z, \quad \partial_x{}^7\partial_y{}^2 z, \quad \partial_x{}^8 z$$

of a system of differential equations in $z = z(x,y)$. The dimension polynomials $H(s)$ and $\omega(s)$ are easily computed as $4s + 12$. In the algebraic case, the degree of $H(s)$ is the dimension of the algebraic set. In the differential case, the degree is the differential type and the leading coefficient is the typical differential dimension. In this example, it means the solutions depend on 4 arbitrary functions of one independent variable.

Open Problem

- **Does a priori knowledge of the differential dimension polynomial (or function) help in computing the characteristic set of an irreducible system of differential equations?**
- **If yes, would it be applicable to compute the characteristic set with respect to another ranking?**

Permutations, Partitions, Young Tableaux

My final topic will be rather sketchy because it involves a lot of notation and technicalities. I want to show you that through the problems of symbolic computations, we can actually obtain **combinatorial results** as a by-product.

Historically speaking, the connection between combinatorics and symbolic computations dates back all the way to the work of Ritt, Levi, Mead and McLenmore, and Grassl. Long before Gröbner basis was developed to solve the membership problem for polynomial ideals, Ritt proved a remarkable theorem called **the Low Power Theorem**, which in some sense gave sufficient conditions that y belongs to the general component of a single differential equation. In 1942, as a tool to generalize this result, **Levi solved the membership problem for specific ordinary differential ideals** $[y^p]$ **and** $[uv]$, where y, u, v are differential indeterminates. Mead and McLenmore (1970) and Mead (1972) gave an algorithm to test membership in differential ideals generated by determinants such as the Wronskian $u'v - vu'$. **Mead derived all possible identities among all** $k \times k$ **subdeterminants of an indeterminate square matrix. These combinatorial relations are obtained by constructing different vector space bases for a subspace of a differential ring and using the fact that the cardinality of all bases is the same.** Grassl (1977, 1981) used the same technique to obtain combinatorial results on the **enumeration of Young's tableaux**.

An Illustration: Partitions and Permutations

To give you just a taste of the connection, I will describe some bijections.

Let w be a nonnegative integer. Recall that a **partition of w into d parts**, or of **degree d**, is a d-tuple $(p_1, \ldots, p_d)$ of *nonnegative* integers p_k with $p_1 \le p_2 \le \cdots \le p_d$ and

$$w = p_1 + \cdots + p_d.$$

Let $P = (P_1, \cdots, P_n)$, where P_i is a partition of w_i of degree d_i, and let

$$w = w_1 + \cdots + w_n.$$

Then P is called an **n-section partition of w** with **signature** $D = [d_1, \cdots, d_n] = \text{sig } P$ and the **weight of P is w**.

Let $E(w, D)$ be the set of all such n-section partitions. An n-section partition P can be represented by a **generalized permutation** Q. For example, if $w = 67$, $D = [4, 3, 3, 2]$, $w_1 = 16, w_2 = 16, w_3 = 20, w_4 = 15$, and $P_1 = (2, 2, 4, 8)$, $P_2 = (3, 5, 8)$, $P_3 = (4, 6, 10)$ and $P_4 = (5, 10)$, then the generalized permutation Q is represented by the 2×8 matrix

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 3 & 4 & 4 \\ 2 & 2 & 4 & 8 & 3 & 5 & 8 & 4 & 6 & 10 & 5 & 10 \end{pmatrix}.$$

Illustration: Generalized Young Tableau

An n-tableau with signature D and weight w (generalized Young tableau) is an n-section $T = (T_1, \cdots, T_n) \in E(w, D)$ such that $d_1, \cdots, d_n$ is non-increasing, and the entries in each column are strictly increasing. For example, $T = P$ is an n-tableau and may be drawn as

2	2	4	8
3	5	8	
4	6	10	
5	10		

Let m be the maximum number appearing in T. The *content* **of** T is the $(m+1)$-tuple $(c_0, c_1, \cdots, c_m)$ where c_i is the number of entries in T which has value i. In the above example, the content of T is

$$\text{con}\,(T) = (0, 0, 2, 1, 2, 2, 0, 2, 0, 2).$$

Now let $D^* = (0, d_1, d_2, \cdots, d_n)$. Let $G(w, D)$ be the set of all ordered pairs of n-tableau (T, T') with $T, T' \in E(w, D)$ and the content of T' is D^*.

Using an **insertion procedure**, Knuth (1970) showed that **there is a bijection between $G(w, D)$ and $E(w, D)$**, the latter viewed as generalized permutations.

Illustration: Ordinary Differential Polynomials

Now let $R = K\{ y_1, y_2, \cdots, y_n \}$ be an ordinary differential polynomial ring. Clearly, R as a vector space over K is generated by all the differential monomials. There is a bijection between the set of all differential monomials and n-sections. The differential monomial corresponding to P is

$$M = y_1^{(2)} y_1^{(2)} y_1^{(4)} y_1^{(8)} y_2^{(3)} y_2^{(5)} y_2^{(8)} y_3^{(4)} y_3^{(6)} y_3^{(10)} y_4^{(5)} y_4^{(10)}.$$

In the case of the differential ideal $[W_n]$, where W_n is the Wronskian of $y_1, y_2, \cdots, y_n$, Mead constructed a vector space basis of R consisting of determinantal products which are in a natural bijection with ordered pairs of Young tableaux of the same shape having n or fewer rows.

Final Comments

Differential polynomials can be used to store information and it is a convenient storage with rich algebraic structures. I have used it for example to output a huge list of underdetermined coefficients as a single differential polynomial by using differential monomials as indexes. This technique has been very useful in computer verification and exploration of two still unsolved conjectures: The **Kolchin-Schmidt Conjecture** and the **Buium Conjecture**. The first has to do with diophantine approximation problems and the second with vanishing theorems in algebraic geometry. The computations, on the other hand, only require knowledge of linear algebra.

Note: The list of References is not yet complete.

References

[1] Axiom, open source version available at `http://wiki.axiom-developer.org`. You can try it on-line or download it.

[2] Kaltofen, E., and Villard, G., On the Complexity of Computing Determinants, Extended Abstract. Computer Mathematics Proc. Fifth Asian Symposium (ASCM 2001) edited by Kiyoshi Shirayanagi and Kazuhiro Yokoyama, Lecture Notes Series on Computing, vol. 9, World Scientilc, Singapore, pages 13–27.
URL:`http://www4.ncsu.edu/~kaltofen/bibliography/01/KaVi01.pdf`.

[3] Kaltofen, E., and Villard, G., On the Complexity of Computing Determinants, *Comput. Complex.* **13**:3–4 (Feb 2005), 91–130. URL:
`http://perso.ens-lyon.fr/gilles.villard/BIBLIOGRAPHIE/PDF/KaVi04.pdf`

[4] Tapia, R. A., and Lanius, C., *Computational Science: Tools for a Changing World*, Rice University.
URL: `http://ceee.rice.edu/Books/CS/chapter5/cost6.html`.

[5] Gardini, L., et al., Bifurcations and transitions to chaos in three-dimensional Lotka-Volterra map. *SIAM J. Appl. Math.* **47**:3 (1987), 455–482.

[6] Sit, W. Y., An Algorithm for Solving Parametric Linear Systems, *J. Symb. Comp.*, **13** (1992), 353–394.