AF:Small: Solving Linear Differential Equations in terms of Special Functions, Project Description

Mark van Hoeij

September 1, 2010 — August 31, 2013

1 Introduction

Linear differential equations with polynomial or rational function coefficients are very common in science. Many scientists use computer algebra systems to solve such equations. Computer algebra systems (Maple, Mathematica, etc.) contain numerous programs to match an equation with an equation in textbooks such as [56]. A big advantage of these programs is that they take much less time than searching the library.

But what if the computer does not find a solution? Does it mean that one has to search elsewhere to find closed form solutions? Or does it mean that no closed form solution exists? The latter would be useful to know because then one can stop searching.

At the moment, the answer to these questions is somewhat subtle. There exist complete algorithms for certain types of solutions (e.g. Liouvillian functions [57]). But there are also classes of well known functions for which current solvers are incomplete. The most important such function is the hypergeometric ${}_2F_1$ function. Equations with ${}_2F_1$-type solutions are very common, particularly in combinatorics and parts of physics [14, 12]. Unfortunately, current computer algebra systems often do not find such solutions, leading users to the incorrect conclusion that closed form solutions are rare.

The PI's longer-term research goal is to develop complete algorithms to find all closed form solutions of (linear) differential equations and recurrence relations. The goals in this project are:

- (top-down approach): Develop general methods to reduce a differential equation to an equation that is easier to solve.
- (bottom-up approach): Design efficient algorithms to solve equations that can not be further reduced with a top-down approach. In par-

ticular for ${}_2F_1$-type solutions, the aim is to divide the problem into a number of special cases, and to develop efficient algorithms for each case.

The combination of these approaches should efficiently find all ${}_2F_1$-type solutions. Note that the top-down methods are not specific to the ${}_2F_1$ hypergeometric function, and are useful regardless of the type of solutions being sought.

Complexity.
It is important to develop efficient algorithms; the research is not completed the moment that an exponential time algorithm has been found. Nowadays, computers are not only used to solve equations, but also to find equations (e.g. [13]). Consequently, equations with coefficients of high degree are now common. If a complete solver is not efficient, then all that the user might receive is a long wait followed by a message that the machine ran out of memory. The theoretical promise of a complete solution is far more valuable when accompanied by an efficient algorithm that can actually deliver on this promise in practice.

1.1 Value to research, education, and society

An important benefit of producing good algorithms is that people benefit from the work even if they are unaware of this research. Their equations will be solved by the computer, and for this there is no need to know what was behind it. Most of the people who benefit from the PI's work will be unaware of it. In fact, in computer algebra, that is precisely how it should be; having a complete algorithm in the computer and obtaining certainty with a click of a button is much preferable over having to search for a closed form solution while never knowing for sure if one exists.

Computer algebra systems are an important part of the infrastructure for research and education. Thus, the value of the algorithms to be developed in this project will increase significantly when these algorithms are incorporated into computer algebra systems. To facilitate this, the PI will make implementations available on the web, and will assist to incorporate the algorithms into commercial as well as free computer algebra systems (see also Section 6 on prior results).

Many branches of science have important impacts on society. Differential equations occur in almost every branch of science, and having closed form solutions is very useful in practical applications. Computer algebra systems

are widely used and are of great value to society. Within computer algebra, differential equations is one of the areas with the highest overall impact.

1.2 Definitions and notations

This subsection introduces terminology that will be helpful to clarify the relation with current and prior work in Section 2, as well as to describe the benefits of the new approaches proposed in Sections 4 and 5.

1. Let L_1 and L_2 be differential equations of the same order with rational function coefficients. We say that L_1 *can be reduced to* L_2 when the solutions of L_1 can be expressed in terms of solutions of L_2. A precise definition of the phrase "can be expressed in terms of" is given in Singer's paper [76]. The definition includes all of the operations that one would normally consider as writing an expression in terms of other expressions, except one. That one operation is composition (i.e. change of variables). See also Item 5 below.

2. If L_1 and L_2 have order 2 and have no Liouvillian solutions[1], then definition in Item 1 turns out to be an equivalence relation, i.e., if L_1 can be reduced to L_2 then L_2 can also be reduced to L_1. In this case we will say that L_1 is *equivalent* to L_2.

3. An algorithm is said to be *equiv-complete* if the following is true: Whenever L_1 and L_2 are equivalent, if the algorithm solves L_2 then it also solves L_1.

4. To design an algorithm that is complete for a certain type of functions, it is necessary that the algorithm is equiv-complete.

5. Being equiv-complete is not a sufficient condition for a complete algorithm, for that, change of variables needs to be included as well.

Given two second order equations L_1 and L_2 with no Liouvillian solutions, there is an implementation called `equiv` available on the PI's website [51] that can decide if L_1 and L_2 are equivalent, and if so, find an operator that maps solutions of L_2 to solutions of L_1.

Example: Consider the following differential equations:

$$L_1(y) = x(x^2-1)^2y'' - (x^2+x+1)y = 0$$

[1] This is not a real restriction since Liouvillian solutions can be found efficiently [44, 57].

and

$$L_2(y) = x(x^2-1)y'' + (3x^2-1)y' + xy = 0.$$

By downloading [51] and calling equiv(L_2,L_1), one finds the following map that sends solutions of L_2 to solutions of L_1

$$y \mapsto x\sqrt{x^2-1}\left((x+1)y' + xy\right). \tag{1}$$

To find the inverse map, just run the same program with L_1, L_2 interchanged.

2 Feasibility and relation to current and prior work

There are numerous algorithms and tools [1, 15, 16, 17, 18, 21, 22, 23, 32, 33, 35, 36, 42, 44, 57, 58, 61, 66, 70, 74, 78, 79, 83, 84, 86] for solving linear differential equations. Each treats a certain type of solution.

Most of the solvers that are currently implemented in computer algebra systems are not equiv-complete (and hence not complete). For the example from Section 1.2, Maple solves L_2 (an equation for elliptic integrals) but not L_1, despite the fact that the map given in equation (1) shows that L_1 and L_2 are equivalent.

Differential equations with closed form solutions are common. For example, Bousquet-Mélou and Mishna [14] examined sequences that count certain lattice-walks, and showed that for 23 of those sequences the generating function is D-finite, which means that it satisfies a differential equation with rational function coefficients. Bostan and Kauers [13] computed differential equations for these generating functions, in particular for the 17 non-algebraic cases. Out of these 17 cases, they could solve only one with current computer algebra systems. However, they all have closed form solutions. The fact that current systems solve some, but not all, equations that have closed form solutions can lead users to the incorrect conclusion that closed form solutions are rare.

This is not to say that currently existing algorithms are not useful. These algorithms save many researchers, including the PI, a lot of time. The proposed project would take much longer if it were not for the fact that some of the required tools are already available. For this project the most important tools are: reduction of order [48], software [43, 51] for finding gauge transformations, hypergeometric [22] exponential [21, 72] and Liouvillian [35, 44, 57, 78, 80] solutions, local data at singular points, subfields of algebraic extensions [45], etc. These tools make it possible to develop and implement the proposed methods in the next 3 years.

2.1 Relation to the PI's current NSF grant

The PI is currently supported by NSF grant 0728853, from September 2007 to August 2010. Three graduate students have received support from this grant. Giles Levy (Ph.D December 2009) and Yongjae Cha (Ph.D expected in August 2010) are working on finding closed form solutions of difference equations, while Quan Yuan (Ph.D expected in 2011) is working on solving differential equations. Another graduate student (Tingting Fang) recently started working with the PI on differential equations. This means that until now (December 2009), two thirds of the effort has been spent on difference equations, and that starting August 2010 most of the effort will go towards differential equations.

The project description for NSF grant 0728853 (available at [49]) described a novel approach for finding closed form solutions of differential and difference equations. This approach has yielded very good results. The majority of the results are on difference equations (see Section 6 for details) because the PI worked with two graduate students on this topic. Relevant for this proposal are the results on differential equations, described below.

For the differential case, complete algorithms have been developed and implemented that can find all ${}_0F_1$ and all ${}_1F_1$-type solutions (this includes Airy, Bessel, Cylinder, Kummer, Laguerre, Whittaker functions, etc.) The first paper [24] includes most of these cases, except the important case where the solution is expressed in terms of a Bessel function composed with the square-root of a rational function. However, this case needed to be treated, because if B_ν is one of the Bessel functions, and if f is a rational function, then $B_\nu(\sqrt{f})$ satisfies a differential equation with rational function coefficients.

It was not difficult to show that the algorithm could be completed for the $B_\nu(\sqrt{f})$-case by solving systems of polynomial equations (with Gröbner basis techniques). However, that would introduce a double exponential complexity, which would be a clear contradiction with the PI's goal of developing efficient algorithms as stated in the paragraph on complexity in Section 1. Thus, an alternative was needed. It turned out that the non-linear polynomial equations could be replaced by linear equations by first solving a number theoretical problem. This has been done with graduate student Quan Yuan. The result is that there now is an efficient algorithm and implementation [50] for all ${}_0F_1$ and all ${}_1F_1$-type solutions, including the Bessel-$\sqrt{f}$ case.

This algorithm [50] could be generalized to higher order equations, to efficiently find all ${}_pF_q$-type solutions except when $p = q + 1$. Unfortunately,

it does not generalize to the ${}_2F_1$ case, which is probably the most important case of all. This means that at the moment there is no efficient (or complete) algorithm for the ${}_2F_1$ case. The PI has used some ad-hoc techniques to compute ${}_2F_1$-type solutions for people working in combinatorics and people working in physics. However, these techniques are neither complete nor efficient, so it has become apparent that new approaches are needed.

To find ${}_2F_1$-type solutions, the PI proposes new top-down and bottom-up approaches, to be outlined in Section 3, with more details in Sections 4 and 5. One graduate student (Quan Yuan) will work on the bottom-up approaches, while the other (Tingting Fang) will work on the top-down approaches.

3 Top-down and bottom-up approaches

To obtain efficient algorithms, the PI proposes several top-down and bottom-up approaches. To illustrate how each of these ideas can contribute to solve differential equations in terms of special functions, a comparison will be made with the classical topic of solving polynomial equations in terms of radicals.

For a polynomial equation of degree n, it is well known that the equation is solvable in terms of radicals if and only if the Galois group is solvable (and that for $n \geq 5$ there exist non-solvable groups). If the group is solvable then Galois theory also shows how, at least in principle, one can find solutions in terms of radicals. So there exists a complete decision procedure. However, such a procedure can be too slow, and so other algorithms are used that are much more efficient. Table I below compares this situation with that of solving differential equations in terms of special functions.

Notation in Table I: P is a polynomial in $\mathbb{Q}[x]$. L is a linear differential equation with rational function coefficients. The key part to finding ${}_2F_1$ type solutions, if they exist, is to first find a rational function $f \in \mathbb{C}(x)$ (and constants a, b, c) for which L is equivalent (item 2 in Section 1.2) to the differential equation for ${}_2F_1(a, b; c; f)$. Let n_f denote the maximum of the degrees of the numerator and denominator of f.

Table I:

Solve P in terms of radicals	Solve L in terms of ${}_2F_1(a,b;c;f)$
Galois theory provides a complete decision procedure.	The PI has ad hoc methods but no complete algorithm exists yet.
Factor [37] to reduce the degree if P is reducible.	Use [48, 42] and factorization [33] to reduce the order when possible.
Solving irreducible P of high degree can be slow.	For large n_f the ad hoc methods become exponentially slow.
Top-down approach: Try to reduce the degree by computing subfields [45, 59]. See Examples 1 and 2 in Section 4.1.	**Top-down approach:** Try to reduce n_f by computing a subfield to which L descends. See Examples 1, 3, and 4.
Bottom-up approach: Develop fast algorithms for $n = 1, 2, 3, \ldots$ that use tables of groups (see [31] for $n \leq 15$).	**Bottom-up approach**: Develop: algorithms that use tables for L with few singularities, special algorithms and tables to treat difficult cases separately, and fast algorithms for small n_f.

The right-hand side of Table I is a brief summary of the ideas that the PI plans to pursue in this proposal. The left-hand side of the table is meant only to clarify the right-hand side (references were given in the table for readers interested in the left-hand side.)

4 The top-down approach

Top-down methods are not specific to any type of solution, such as ${}_2F_1$. The aim is not to try to find a particular type of solution (that will be the aim in Section 5), instead, the aim is to reduce the equation to one that is easier to solve. Existing top-down algorithms reduce the order of the equation in various ways; the PI has developed and implemented a differential factorizer [33], as well as other algorithms to reduce the order [42, 48] (with NSF support). The existence of these order-reduction algorithms motivates the PI to work on second order equations first, because progress there will also help solve higher-order equations.

In contrast with existing top-down algorithms, the proposed top-down methods will not to reduce the order, instead, they will to reduce the de-

grees[2] of the problem for the bottom-up algorithms proposed in Section 5.

At the moment, the PI's ad-hoc techniques for ${}_2F_1$-solutions are still incomplete and have running times that are at least exponential in n_f. The goal in Section 5 will be to solve both problems, but until then, a reduction in n_f obtained by the proposed top-down techniques means an exponential improvement in running time.

The top-down approaches have an additional benefit, in that they not only help to solve in terms of the special functions the PI is currently considering, but will also help when new functions are added to this in the near future.

4.1 2-descent

The first of the proposed top-down approaches, called 2-descent, will be illustrated with 4 examples.

Example 1 Descent to an index-2 subfield, trivial case. *Consider the following polynomial resp. differential equation.*

$$P_1(x) = x^8 - 8x^2 + 9 = 0, \quad L_1(y) = (x^5 - x)y'' + (6x^4 + 1)y' + 4x^3y = 0$$

Substituting[3] *(change of variables)* $x \mapsto \sqrt{t}$ *turns these equations into*

$$P_2(x) = t^4 - 8t + 9 = 0, \quad L_2(y) = (t^2 - 1)y'' + \frac{7}{2}ty' + y = 0$$

which have lower degree resp. lower n_f*.*

The equations in Example 1 are easy to solve with a computer algebra system, and the simplification $x \mapsto \sqrt{t}$ was easy to find. The number field $\mathbb{Q}[x]/(P_1)$ contains an index-2 subfield isomorphic to $\mathbb{Q}[t]/(P_2)$. Likewise, $\mathbb{C}(t) = \mathbb{C}(x^2)$ is a subfield of $\mathbb{C}(x)$ with index 2, and for this reason the reduction from an equation L_1 with coefficients in $\mathbb{C}(x)$ to an equation L_2 with coefficients in $\mathbb{C}(t)$ will be called *2-descent*.

Example 2 Compute an index-2 subfield. *Consider the equation*

$$P_1(x) = x^8 - 3x^6 - 10x^5 - x^4 - 20x^3 - 12x^2 + 16 = 0.$$

If x *satisfies* P_1 *then* $t := x + 2/x$ *satisfies* $P_2(t) = t^4 + 2t + 2 = 0$*. This way, a degree-8 equation* P_1 *is reduced to a degree-4 equation* P_2*.*

[2] the numbers n_f and n_r in Section 5

[3] The $'$ means $\mathrm{d}/\mathrm{d}x$ in L_1 and $\mathrm{d}/\mathrm{d}t$ in L_2.

The reduction in Example 2 was less obvious than the one in Example 1. However, in both cases the reduction can be interpreted as the computation of a subfield. There are complete algorithms [45, 59] to compute subfields, and hence the type of reduction $P_1 \to P_2$ from Examples 1 and 2 can always be found whenever it exists. Moreover, computing subfields is a very efficient way to solve P_1. It is therefore reasonable to expect that finding such a reduction (when it exists) should also be possible for linear differential equations, and that this will improve the overall performance.

Example 3 2-descent, easy case. *Consider the equation*

$$L_1(y) = (x^2-2)x^5y'' - 4x^4y' - (x^2+2)(x^2-2)^3y = 0.$$

The map $x \mapsto -x$ that lead to 2-descent in Example 1 is an example of a Möbius transformation $x \mapsto (ax+b)/(cx+d)$. One can check if there exists a Möbius transformation $\sigma \neq \mathrm{id}$ that fixes the singularity structure of L. In this example one finds $\sigma : x \mapsto 2/x$. The fixed field of σ is $\mathbb{C}(t)$ where $t := x + 2/x$. That corresponds to $x = (t + \sqrt{t^2-8})/2$. Making this change of variables leads to $L_2(y) = y'' - ty = 0$.

Example 3 is easy to solve because the Möbius transformation σ preserves not only the singularity structure of L_1, but also L_1 itself. That means that the change of variables $x = (t + \sqrt{t^2-8})/2$ will produce an equation L_2 without the square root.

Writing an algorithm that performs the simplification illustrated in Example 3 would be very easy if one only treated the case where L_1 is invariant under σ. However, such an algorithm would be of limited value because it would not be equiv-complete (defined in Section 2.1). To obtain a complete algorithm for 2-descent, one must do this: Compute, if it exists, a Möbius transformation $\sigma \neq \mathrm{id}$ that preserves the singularity structure of L_1. Then find, if it exists, an *equivalent* equation L_2 that is invariant under σ.

Example 4 2-descent, general case. *Consider the equation L_1 from the example in Section 1.2. Its singularity structure does not change under $\sigma : x \mapsto -x$, but unlike Example 1 above, L_1 itself is not invariant under σ, which means that the equation one obtains through the change of variables $x \mapsto \sqrt{t}$ will contain $\sqrt{t}$ and will thus not be simpler than L_1. To apply 2-descent for L_1, one has to find an equation L_2 that is invariant under σ and equivalent to L_1. Such L_2 is given in the example in Section 1.2. One can then apply $x \mapsto \sqrt{t}$ to L_2 and obtain the equation $x(x-1)y'' + (2x-1)y' + y/4 = 0$. In this example, 2-descent reduced n_f from 2 to 1.*

The PI has done theoretical work on descent in [47] with NSF support. The next task is to develop an efficient algorithm, which will be done jointly with graduate student Tingting Fang.

4.2 3-descent

The first problem of descent is to find the correct subfield of $\mathbb{C}(x)$ to which one can descend. If the index of that subfield is 2, then this subfield must be the fixed field of some automorphism of $\mathbb{C}(x)$, i.e. the fixed field of some Möbius transformation. So this subfield can be computed by determining all Möbius transformations that fix the singularity structure.

Descent to an index-3 subfield is trickier because a degree-3 field extension need not be a Galois extension, in which case this index-3 subfield will not be the fixed field of a Möbius transformation.

Let $f \in \mathbb{C}(x)$ with $n_f = 3$, so $\mathbb{C}(f)$ is a subfield of $\mathbb{C}(x)$ of index 3. If L_1 is equivalent to a equation L_2 that descends to $\mathbb{C}(f)$, then how would one find this subfield $\mathbb{C}(f)$ when only L_1 is given? To answer that, consider the normal closure of $\mathbb{C}(x)$ over $\mathbb{C}(f)$. It has the form $\mathbb{C}(x, \sqrt{D})$ where D is a polynomial of degree ≤ 4. To find $\mathbb{C}(f)$ one has to find this field $\mathbb{C}(x, \sqrt{D})$, then compute an S_3-group of automorphisms of this field that fix the singularity structure, and then compute the fixed field.

If the degree of D is ≤ 2 then $\mathbb{C}(x, \sqrt{D})$ is isomorphic to a rational function field. If the degree is 3 or 4, then $\mathbb{C}(x, \sqrt{D})$ is an elliptic function field. Such a field has only finitely many automorphisms of order 3 so it should be possible to find the above mentioned S_3-group. The main question that needs to be studied is how to compute D efficiently.

Descent techniques form a mix of algorithmic issues and beautiful classical mathematics such as elliptic curves. As such, this topic provides excellent opportunities for graduate students. The PI plans to work with one graduate student on descent techniques and other top-down techniques that are yet to be discovered. The PI will work with another graduate student on bottom-up techniques, the topic of the next section.

5 Bottom-up approach

A definition is needed to describe the problems, and the PI's approach to handle them: A singularity p of L is called an *apparent singularity* if L is equivalent (item 2 in Section 1.2) to another equation for which p becomes a regular point. Otherwise p is called a true singularity. Let n_a denote the number of apparent singularities, and n_r the number of true singularities.

To obtain a complete algorithm, it is necessary to design the algorithms to be equiv-complete (defined in Section 1.2). That implies that to find solutions in terms of ${}_2F_1(a,b;c;f)$ one has to construct f (and the constants a,b,c) without making use of the apparent singularities. Thus, f must be constructed from the n_r true singularities.

If L has true irregular singularities (singularities that stay irregular under equivalence), and L has order 2, then any ${}_pF_q$-type solution must be of the type ${}_0F_1$ or ${}_1F_1$. The PI now has complete algorithms for these cases (supported by the PI's current NSF grant). So from here on, we can assume that the n_r true singularities are regular singularities. Likewise, one can also assume that there are no Liouvillian solutions (Such solutions can be found with [57, 44].) With L having rational function coefficients, one can usually assume that f is a rational function, except in certain very special cases that shall be treated separately. Then let n_f denote the degree of f, the maximum of the degree of the numerator and the degree of the denominator.

At the moment, there does not exist an algorithm to determine a,b,c and n_f. Instead, the PI currently uses heuristic guesses for a,b,c and has experimentally found formulas for n_f. Proving such formulas would be a key step towards obtaining a complete decision procedure. However, the initial the emphasis in the bottom-up approach is efficiency; treating as many as possible cases as quickly as possible.

One of the first things to do is to complete the PI's ad-hoc methods to the case where n_f is small, say $n_f \leq 5$. This 5 may well change during the research. The case $n_f = 1$ is trivial because n_r must be 3 in this case, and a degree-1 rational function (a Möbius transformation) is determined by 3 points. Conversely, if $n_r = 3$ then one can choose $n_f = 1$ as in [74]. Each of the cases $n_f = 2,3,4,5$ will be split into a finite number of sub-cases, with separate algorithms for each.

5.1 Four singularities

After $n_f \leq 5$, the next case to consider is $n_r = 4$. Although handling $n_f \leq 5$ does not fully cover the case $n_r = 4$, it does cover most of the $n_r = 4$ case, so the idea is now simply this: Classify all equations with all of the following properties: $n_r = 4$, $n_f > 5$, and f does not allow 2-descent or 3-descent (This means that in the field extension $\mathbb{C}(f) \subset \mathbb{C}(x)$ there is no subfield of index 2 or 3.)

If $n_r = 4$ and $n_a = 0$ then L can be solved in terms of Heun functions. In this case, the problem of finding ${}_2F_1$-type solutions can be viewed as the problem of deciding which Heun functions can be expressed in terms

of ${}_2F_1$-functions. This problem was treated in [65]. For our purposes this needs to be extended to the case $n_a > 0$. Nevertheless, this and many other results from the literature will help with the above mentioned classification problem.

Consider a Möbius transformation $\sigma : x \mapsto (ax + b)/(cx + d)$, where $ad - bc \neq 0$. If σ, viewed as a change of variables, sends L_1 to L_2, then solving L_1 is the same problem as solving L_2 (after all, Möbius transformations are invertible). Because of this, the singularity structure should be considered only up to Möbius transformations.

Four points in $P^1(\mathbb{C}) = \mathbb{C} \bigcup \{\infty\}$ can be classified (up to Möbius transformations) with the cross-ratio. So the classification problem is then: find a complete list of cross-ratio's for which there is a corresponding equation with $n_r = 4$, $n_f > 5$, and no 2 or 3 descent. For each such cross-ratio, store the corresponding a, b, c, f in a table. Once the table is completed, there will be a complete solver for the $n_r = 4$ case. For this table to be finite, it is necessary to exclude small n_f from the table (The PI proposed to treat $n_f \leq 5$ separately, but this cut-off point was arbitrary and may change during the research.)

If all singularities $p_1, \ldots, p_4$ have the same exponent-difference modulo the integers, then they can be treated as a set of unordered points, which are classified with the j-invariant of $y^2 - (x - p_1) \cdots (x - p_4)$.

Like the cross-ratio for $n_r = 4$, there exist similar convenient invariants that classify 5 or 6 points up to Möbius transformations.

5.2 What comes next

For almost every linear differential equation one might realistically encounter in research, the combination of top-down and bottom-up approaches should find closed form solutions whenever they exist. But what if no closed form solution for an equation L can be found? The user would want to know if it has been proven that no such solutions exist. Such proofs can be obtained using the monodromy group, see [60] for an example. How to construct such proofs algorithmically will be a topic of research after the top-down and bottom-up approaches have been developed.

6 Results from prior NSF support.

The following is an overview of research supported by the PI's prior NSF grants active during 2004–2009 (for details on those grants see Subsection 6.1).

1. Recurrence relations

   (a) Solutions of Second Order Recurrence Relations.
   Giles Levy, one of the PI's graduate students that was supported by NSF grant 0728853, gives three new algorithms in his Ph.D thesis (December 2009, [63]). The first algorithm decides if a recurrence relation can be solved in terms of the sequence $u(n) = {}_2F_1(a+n, b; c; z)$ for some constants a, b, c, z.
   Next, an algorithm for Liouvillian solutions is given. This algorithm treats only order 2, but it is the fastest Liouvillian solver for order 2. (The combinatorial problem has been reduced to a single case.) The third algorithm checks if an equation can be reduced to a second order equation whose solution appears in Sloane's On-Line Encyclopedia of Integer Sequences [82].
   Each of these algorithms has the property given in Item 3 in Section 1.2. That makes them very strong in practice. Each of the three algorithms solves many of the equations in Sloane's On-Line Encyclopedia [82], including a large number of equations for which the On-Line Encyclopedia does not (yet) have a solution. The implementation can be downloaded from [63].

   (b) The PI and graduate student Yongjae Cha have implemented algorithms to compute data for recurrence relations that is invariant under the equivalence from Item 2 in Section 1.2. This way, solvers such as those in Levy's thesis [63] can now be developed quickly in a systematic way. Examples currently implemented include Liouvillian functions [52], Bessel, and Whittaker (in preparation). The paper for the Liouvillian case [52] won an award at the ISSAC'2009 conference for best student co-authored paper.

   (c) In [22] the PI, joint with T. Cluzeau, developed an efficient algorithm to compute hypergeometric solutions of linear recurrence relations, that avoids computing splitting fields which was a bottleneck in Petkovšek's algorithm (see [69]). An implementation by the PI of this algorithm is available.

   (d) In [2] an algorithm was developed to desingularize recurrence relations whenever possible.

2. Solving linear differential equations.

   (a) In [24, 50] a complete algorithm to solve all second order equations with ${}_0F_1$ and ${}_1F_1$ type solutions has been developed. See

Section 2.1 for more details.

(b) The Kovacic algorithm [57] is a famous algorithm for finding Liouvillian solutions of linear second order differential equations. With the use of Klein's theorem, the PI and J.A. Weil could give a more efficient algorithm [44] that produces more compact solutions. An implementation of this algorithm is available [57].

(c) A new algorithm for computing exponential solutions was developed in [21]. One of the novel ideas in this algorithm is to combine local data in characteristic 0 (generalized exponents) with global mod p data (the p-curvature).

(d) In [17] equations with doubly periodic coefficients were treated.

3. Factoring polynomials.
The PI developed a new algorithm [37] to factor polynomials with rational number coefficients, specifically, the combinatorial problem appearing in Zassenhaus' algorithm is solved efficiently. This algorithm is a significant practical improvement [64]. It was soon incorporated into computer algebra systems such as Maple, Magma, NTL, Pari, and MuPAD. These implementations benefit indirectly from NSF support, because those implementations are assisted by the paper as well as the PI's implementation on the web, both of which were produced with NSF support.

   More recently, the PI and graduate student A. Novocin have proven a new complexity result [67, 53] for this factoring algorithm, the first improvement in the complexity of factoring in $\mathbb{Q}[x]$ since the 80's. Moreover, this complexity analysis also lead to practical improvements that will make the algorithm faster on the most common cases (specifically, the cases that have one large irreducible factor and zero or more small irreducible factors).

4. Evaluating Riemann theta functions, Riemann matrices.
The following algorithms were developed in joint work with Bernard Deconinck: monodromy, homology, differentials, periodmatrix, and an algorithm to compute Riemann Theta functions [26]. See also NSF nugget [68].

5. Order reduction for differential operators.
The PI has developed several algorithms for reducing differential equations to differential equations of lower order. Besides factoring [33],

the PI has also used other constructions (symmetric product [42], symmetric power, or a symmetric power after a gauge transformation [48]) to develop and implement algorithms for reducing the order. To do these reductions efficiently required the development of an algorithm for conics [46], as well as answers to a number of theoretical questions [47] on descent of differential operators.

The PI has also developed an algorithm to compute gauge transformations. This algorithm is also useful for computing ladder operators in physics, and is available at [43].

6. Modular GCD algorithm.
   In computations involving algebraic extensions, a bottleneck in the computation is often GCD computations. To remedy this, the PI wrote two joint papers [40], [41] with M. Monagan on modular algorithms for GCD computation, one for number fields presented with multiple extensions, and one for function fields.

6.1 List of prior NSF grants active during 2004 – 2009

Title: Algorithms for Linear Differential Equations and Algebraic Functions. NSF 0098034, 09/15/01 – 08/31/04, $152,585

Title: Simplifying Algebraic Numbers and Algebraic Functions.
NSF 0511544, 09/01/05 – 08/31/08, $89,999

Title: Closed Form Solutions for Linear Differential and Difference Equations. NSF 0728853, 09/01/07 – 08/31/10, $275,000

Supported Graduate students:
The grants listed above have supported Andrew Novocin (Ph.D April 2008, currently a postdoc at ENS de Lyon), Giles Levy (Ph.D December 2009), Yongjae Cha (Ph.D expected in August 2010), and Quan Yuan (Ph.D expected in 2011).

Papers supported:
Journal publications: [2, 4, 10, 21, 22, 26, 37, 46, 47]
Refereed conference publications: [17, 24, 30, 40, 41, 42, 44, 48, 52]
Preprints submitted for publication: [53, 54]
Other preprints: [27, 39, 38, 45]
The PI's papers are available at: www.math.fsu.edu/~hoeij/papers.html

References

[1] S.A. Abramov, *Rational solutions of linear differential and difference equations with polynomial coefficients*, USSR Comput. Maths. Math. Phys. 29, 7-12 (translated from Zh. vychisl. mat. fiz. 29, 1611-1620) (1989).

[2] S.A. Abramov, M. Barkatou, and M. van Hoeij, *Apparent Singularities of Linear Difference Equations with Polynomial Coefficients*, AAECC, **17**, 117-133 (2006).

[3] S.A. Abramov and M. van Hoeij, *A method for the Integration of Solutions of Ore Equations* ISSAC '97 Proceedings, 172-175 (1997).

[4] S.A. Abramov and M. van Hoeij, *Set of Poles of Solutions of Linear Difference Equations with Polynomial Coefficients*, Computational Mathematics and Mathematical Physics, **43**, No. 1, 57-62 (2003).

[5] M. Abramowitz and I.A. Stegun, *Handbook of Mathematical Functions*, Dover Publications, New York, (1972).

[6] G. Andrews, R. Askey and R. Roy, *Special Functions*, Encyclopedia of Mathematics and its Applications, **71**, Cambridge University Press, (1999).

[7] F. Baldassarri and B. Dwork, *Differential Equations with Algebraic Solutions*, American Journal of Mathematics, **101**, 42-76 (1979).

[8] M. A. Barkatou, *On Rational Solutions of Systems of Linear Differential Equations*, J. Symbolic Computation, **28** 547-567 (1999).

[9] K. Belabas, *A relative van Hoeij algorithm over number fields*, J. Symbolic Computation, **37**, 641-668 (2004).

[10] K. Belabas, J. Klüners, M. van Hoeij, and A. Steel *Factoring polynomials over global fields*, Journal de Théorie des Nombres de Bordeaux, **21**, 15-39 (2009).

[11] M. Berry, *Why are special functions special?* Physics Today, 54, no.4, 11-12, (2001). http://www.physicstoday.com/pt/vol-54/iss-4/p11.html

[12] A. Bostan, S. Boukraa, S. Hassani, J.-M. Maillard, J.-A. Weil, N. Zenine, *Globally nilpotent differential operators and the square Ising model*, preprint arXiv:0812.4931 (2008).

[13] A. Bostan, M. Kauers, *Automatic Classification of Restricted Lattice Walks*, preprint arXiv:0811.2899 (2008).

[14] M. Bousquet-Mélou, M. Mishna, *Walks with small steps in the quarter plane*, preprint arXiv:0810.4387 (2008).

[15] M. Bronstein, *On Solutions of Linear Differential Equations in their Coefficient Field*, J. of Symbolic Computation, **13**, 413-439 (1992).

[16] M. Bronstein and S. Lafaille, *Solutions of linear ordinary differential equations in terms of special functions*, Proceedings of ISSAC'02, Lille, ACM Press, 23-28 (2002).

[17] R. Burger, M. van Hoeij and G. Labahn, *Closed Form Solutions of Linear Odes having Elliptic Function Coefficients*, ISSAC'04 Proceedings, 58-64, (2004).

[18] L. Chan, E.S. Cheb-Terrab, *Non Liouvillian solutions for second order linear ODEs*, Proceedings of ISSAC'04, Santander, Spain (2004).

[19] E.S. Cheb-Terrab, *Solutions for the General, Confluent and Bi-Confluent Heun equations and their connection with Abel equations*, Journal of Physics A: Mathematical and General, **37**, 9923-9949 (2004).

[20] E.S. Cheb-Terrab, A.D. Roche, *Hypergeometric solutions for third order linear ODEs*, preprint arXiv:0803.3474 (2008).

[21] T. Cluzeau, M. van Hoeij, *A Modular Algorithm to Compute the Exponential Solutions of a Linear Differential Operator*, J. Symbolic Computation, **38**, 1043-1076 (2004).

[22] T. Cluzeau, M. van Hoeij, *Computing Hypergeometric Solutions of Linear Recurrence Equations*, AAECC, **17**, 83-115 (2006).

[23] E. Compoint, J.A. Weil, *Absolute reducibility of differential operators and Galois groups*, J. Algebra, **275**, 77-105, (2004).

[24] R. Debeerst, M. van Hoeij, W. Koepf, *Solving Differential Equations in Terms of Bessel Functions*, ISSAC'08 Proceedings, 39-46, (2008).

[25] B. Deconinck and M. van Hoeij, *Computing Riemann matrices of algebraic curves*. PhysicaD, 152, 28-46 (2001).

[26] B. Deconinck, M. Heil, A. Bobenko, M. van Hoeij, M. Schmies. *Computing Riemann Theta Functions*, Math. Comp., **73**, 1417-1442 (2004).

[27] W.N. Everitt, D.J. Smith and M. van Hoeij, *The Fourth-Order Type Linear Ordinary Differential Equations*, (2006). `http://arxiv.org/abs/math.CA/0603516`

[28] M. Foupouagnigni, W. Koepf and A. Ronveaux, *On solutions of fourth-order differential equations satisfied by some classes of orthogonal polynomials*, J. Comput. Appl. Math., **162**, 299-326 (2004).

[29] A.R. Forsyth, *Differential Equations I–VI*, Cambridge University Press, Cambridge, England (1906).

[30] A. Galligo and M. van Hoeij, *Approximate Bivariate Factorization, a Geometric Viewpoint*, SNC'2007 Proceedings, 1-10 (2007).

[31] K. Geissler, J. Klüners, *Galois Group Computation for Rational Polynomials*, J.Symb.Comput., **30**, 653-674, (2000).

[32] P. Hendriks and M. Singer, *Solving Difference Equations in Finite Terms*. J. Symbolic Computation, **27**, 239-259 (1999).

[33] M. van Hoeij, *Factorization of Differential Operators with Rational Functions Coefficients*, J. Symbolic Computation, **24**, 537-561 (1997).

[34] M. van Hoeij and J-A. Weil, *An algorithm for computing invariants of differential Galois groups*. J. Pure Appl. Algebra, 117&118, 353-379 (1997).

[35] M. van Hoeij, J.F. Ragot, F. Ulmer and J.A. Weil, *Liouvillian solutions of linear differential equations of order three and higher*. J. Symbolic Computation, **28**, 589-609 (1999).

[36] M. van Hoeij, *Finite Singularities and Hypergeometric Solutions of Linear Recurrence Equations*, J. Pure Appl. Algebra, 139, 109-131 (1999).

[37] M. van Hoeij, *Factoring polynomials and the knapsack problem*, J. of Number Theory, 95, 167-189, (2002).

[38] M. van Hoeij, *A conjecture in the problem of rational definite summation*, `http://arxiv.org/abs/math.CO/0210158`

[39] M. van Hoeij, *An algorithm for computing the Weierstrass normal form of hyperelliptic curves*, `http://arxiv.org/abs/math.AG/0203130`

[40] M. van Hoeij and M. Monagan, *A Modular GCD algorithm over Number Fields presented with Multiple Extensions*, ISSAC'02 Proceedings, (2002).

[41] M. van Hoeij and M. Monagan, *Algorithms for Polynomial GCD Computation over Algebraic Function Fields*. ISSAC'04 Proceedings, 297-304, (2004).

[42] M. van Hoeij, *Decomposing a 4'th order linear differential equation as a symmetric product*, Banach Center Publications, **58**, 89-96, (2002).

[43] M. van Hoeij, *Software for computing gauge transformations* (2004). `www.math.fsu.edu/~hoeij/files/Hom`

[44] M. van Hoeij and J.A. Weil, *Solving Second Order Linear Differential Equations with Klein's Theorem*, ISSAC'05 Proceedings, 340-347, (2005). Implementation available at `www.unilim.fr/pages_perso/jacques-arthur.weil/issac05/`

[45] M. van Hoeij and J. Klüners, *Generating Subfields*, preprint (2005). `www.math.fsu.edu/~hoeij/papers.html`

[46] M. van Hoeij, J. Cremona, *Solving conics over function fields*, Journal de Théorie des Nombres de Bordeaux, 18, p. 595-606 (2006).

[47] M. van Hoeij, M. van der Put, *Descent for differential modules and skew fields*. Journal of Algebra, **296**, 18-55 (2006).

[48] M. van Hoeij, *Solving Third Order Linear Differential Equations in Terms of Second Order Equations*, ISSAC'07 Proceedings, 355-360, (2007).

[49] M. van Hoeij, *Closed Form Solutions for Linear Differential and Difference Equations*, Project Description of NSF grant 0728853, Sept. 2007 – Aug. 2010.

[50] M. van Hoeij, R. Debeerst, Q. Yuan, *Implementation for finding 0F1 and 1F1 type solutions*, `www.math.fsu.edu/~hoeij/files/0F1_1F1`

[51] M van Hoeij, *Implementation for finding equivalence map*, `www.math.fsu.edu/~hoeij/files/equiv`

[52] Y. Cha, M. van Hoeij, *Liouvillian Solutions of Irreducible Linear Difference Equations*, ISSAC'2009 Proceedings, 87-93 (2009).

[53] M. van Hoeij, A. Novocin, *Gradual sub-lattice reduction and a new complexity for factoring polynomials*, accepted for LATIN 2010.

[54] S. Abramov, M. Barkatou, M. van Hoeij, M. Petkovšek, *Subanalytic Solutions of Linear Difference Equations and Multidimensional Hypergeometric Sequences*, submitted to JSC.

[55] E. Ince, *Ordinary Differential Equations*, Dover Publications, New York, (1956).

[56] Kamke E. 1959 *Differentialgleichungen: Lösungsmethoden und Lösungen.* Chelsea Publishing Co, New York.

[57] J. Kovacic, *An algorithm for solving second order linear homogeneous equations*, J. Symbolic Computation, **2**, p. 3-43 (1986).

[58] M. Kauers, *Algorithms for Nonlinear Higher Order Difference Equations*, Doctoral Thesis, RISC Linz, (2005).

[59] J. Klüners, M. Pohst, *On Computing Subfields*, J.Symb.Comput., **24**, 385-397, (1997).

[60] D. Krammer, *An example of an arithmetic Fuchsian group*, J. reine angew. Math. **473**, 69-85 (1996).

[61] G. Labahn, *Solving Linear Differential Equations in Maple*, MapleTech 2(1) 20-28, (1995).

[62] H.Q. Le, *SumTools Package*, http://algo.inria.fr/le/SumTools.html

[63] G. Levy, *Solutions of Second Order Recurrence Relations*, Ph.D thesis, (2009). text and implementation available at
`http://www.math.fsu.edu/~hoeij/glevy`

[64] *Magma Computer Algebra. Factorization.* In Magma help document http://magma.maths.usyd.edu.au/magma/htmlhelp/text560.htm

[65] R.S. Maier, *On reducing the Heun equation to the hypergeometric equation*, J. Differential Equations, **213**, 171-203 (2005).

[66] K. A. Nguyen, M. van der Put, *Solving linear differential equations*, preprint, (2006).

[67] A. Novocin, *Factoring Univariate Polynomials over the Rationals*, Ph.D thesis, (2008).

[68] NSF nugget, http://www.math.fsu.edu/~hoeij/papers/computingtheta

[69] M. Petkovšek. *Hypergeometric solutions of linear recurrences with polynomial coefficients.* J. Symbolic Computation, **14**, 243-264, (1992).

[70] A. C. Person, *Solving Homogeneous Linear Differential Equations of Order 4 in Terms of Equations of Smaller Order*, PhD thesis, www.lib.ncsu.edu/theses/available/etd-08062002-104315/ (2002).

[71] M. van der Put, *Galois Theory of Differential Equations, Algebraic Groups and Lie Algebras*, J. Symbolic Computation **28**, 441-472 (1999).

[72] M. van der Put, M.F. Singer, *Galois Theory of linear Differential Equations*, Grundlehren der mathematischen Wissenschaften, **328**, Springer (2003).

[73] M. van der Put and M. Singer, *Galois Theory of Difference Equations*, Lecture Notes in Mathematics, **1666**, Springer-Verlag, (1997).

[74] A.V. Shanin and R.V. Craster, *Removing false singular points as a method of solving ordinary differential equations*, European Journal of Applied Mathematics, **13**, 617-639 (2002).

[75] M.F. Singer, *Liouvillian Solutions of n-th order Homogeneous Linear Differential Equations*, American Journal of Mathematics, **103**, 661-682 (1981).

[76] M.F. Singer, *Solving Homogeneous Linear Differential Equations in Terms of Second Order Linear Differential Equations*, American J. of Math., **107**, 663-696, (1985).

[77] M. F. Singer, *Algebraic Relations Among Solutions of Linear Differential Equations: Fano's Theorem*, Am. J. of Math., **110**, 115-143, (1988).

[78] M.F. Singer, *Liouvillian Solutions of Linear Differential Equations with Liouvillian Coefficients*, J. Symbolic Computation, **11**, 251-273 (1991).

[79] M.F. Singer and F. Ulmer, *Liouvillian and algebraic solutions of second and third order linear differential equations.* J. Symbolic Computation, **16**, 37-73 (1993).

[80] M.F. Singer and F. Ulmer, *Linear Differential Equations and Products of Linear Forms*, J. of Pure and Applied Algebra, **117**, 549-564 (1997).

[81] S.Y. Slavyanov and W. Lay, *Special Functions, A Unified Theory Based on Singularities*, Oxford Mathematical Monographs (2000).

[82] N. Sloane, *The On-Line Encyclopedia of Integer Sequences*, `http://www.research.att.com/~njas/sequences`

[83] F. Ulmer, *Liouvillian solutions of third order differential equations*, J. Symb. Comp., **36**, 855-889, (2003).

[84] F. Ulmer and J.A. Weil, *A Note on Kovacic's Algorithm*, Journal of Symbolic Computation, **22** 179-200 (1996).

[85] Vidunas, R: *Algebraic transformations of Gauss hypergeometric functions*, http://arxiv.org/abs/math.CA/0408269 (2004).

[86] B. Willis, *An extensible differential equation solver for computer algebra*, SIGSAM, March (2001).