# Closed Form Solutions for Linear Differential and Difference Equations, Project Description

Mark van Hoeij

September 1, 2007 — August 31, 2010

## 1 Introduction

Many scientists use computer algebra systems to solve linear differential or difference equations. These programs check if an equation can be matched with an equation in textbooks such as [58]. 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.

To examine these questions take the following example.

$$y'' \; - \; \frac{8x^5+8x^3-42x}{4x^4-12x^2+3}\; y' \; + \; \frac{(64+8n)x^4-(96+24n)x^2+6n}{4x^4-12x^2+3}\; y \; = \; 0$$

This equation has more singularities than typical textbook equations. So it not surprising that pattern matching techniques fail here, and that current computer algebra systems do not solve this equation. It does, however, have closed form solutions [36]. Many equations in research [29, 30] have closed form solutions and yet are not solved by computer algebra systems.

The goal in this project is to develop a *decision procedure* (a provably complete algorithm) for the following problem: Given any linear differential or difference equation with rational function coefficients, decide if it is solvable in closed form, and if so, find its closed form solutions.

Intuitively, a function is in closed form if it can be expressed in terms of well known special functions. Of course, if two functions can be written in terms of special functions, then so can their sum, product, composition,

their derivatives (shifts in the difference case), etc. Therefore, the class of closed form functions must be closed under at least those operations.

Closure properties are of crucial importance; if even one of them is missing then it would be easy to write down equations that are solvable in closed form, but that are not solved by the algorithm. Current algorithms for finding special function solutions have no such closure properties, not even addition, so they are far from complete. Computers currently only find special function solutions of equations that are close to textbook form.

The intuitive definition of closed form is made precise as follows.

**Definition of closed form**.
Consider a set of functions $F$ and a set of operations $O$. A function is in $(F, O)$-closed form if it is written in terms of the functions in $F$, using the operations in $O$.

For example, if $F = \{\mathbb{C}(x), \exp, \log\}$ and $O = \{$field operations, algebraic extensions, composition, and differentiation$\}$ then the $(F, O)$-closed functions are the *elementary functions* [13, 76]. If one adds integration, so $O := \{+, -, \cdot, /,$ algebraic extensions, $\circ, ', \text{and } \int \mathrm{d}x\}$ then the $(F, O)$-closed functions are the *Liouvillian functions* [43, 52, 61, 83, 85].

In this project $F$ will consist of $\{\mathbb{C}(x)$, exp, log, Airy, Bessel, Kummer, Whittaker, and ${}_2F_1$-hypergeometric functions$\}$. The set of operations $O$ will be the same as in the Liouvillian case, and $(F, O)$-closed form will simply be called *closed form*. Note that one may add other well known special functions such as sin, cos, Cylinder, Hermite, Laguerre, ${}_0F_1$, ${}_1F_1$, or Legendre functions to the collection $F$ (or remove Airy, Bessel and Kummer functions) without changing the notion of closed form, because these functions can be expressed in terms of other functions already listed in $F$.

The mathematical property that describes the special functions selected in $F$ is that they satisfy a rigid [59] second order differential equation. This is not an arbitrarily chosen set of special functions; these are precisely the special functions about which a great deal of useful information is known, see for example Abramowitz and Stegun [6]. That is why solutions written in terms of these special functions are so useful (for more on this see Section 1 in [67]).

An analogous definition for closed form sequences can be given in the difference case as well (the analogue of rigid equations can be defined using local data at infinity [65] and the PI's notion of finite singularities [23, 44]).

**Relation to prior work**.
It is not the PI's goal to gradually solve more and more equations compared

to prior algorithms by developing new pattern matching techniques or some ad hoc extensions of existing methods. That would be a task that could be pursued indefinitely. Instead, the goal is to develop a *decision procedure*, which means an algorithm that is provably complete for the entire class of closed form functions. Once this algorithm has been developed and its completeness has been proven, then the work is done. If closed form solutions exist, they will be found, and if an equation is not solved, it means that it provably does not have closed form solutions.

**Current status and feasibility of the project**.
It is easy to find and solve some equations that are not solved by current computer algebra systems. Nor would it be hard to increase the capabilities of existing algorithms by making some extensions ([92] is specifically designed to be extendible). However, it is an entirely different matter to design an algorithm that is complete on the entire class of closed form functions. For this task, completeness proofs are a major part of the work.

The PI has a structure theorem for the Bessel case that has most of the desired closure properties. To complete it, more work is need, particularly for the ${}_2F_1$ case. See Section 2.2 for technical details. The main technical points are these: To prove completeness for the Bessel case one can use asymptotic analysis as well as analytic continuation, while for the ${}_2F_1$ case one can only use the latter. Even so, it is still possible to obtain enough information from this to prove the structure theorems needed to prove completeness. The reason this will work is that the special functions considered are rigid [59], which means that their global behavior (their behavior under analytic continuation) is determined by their local asymptotic behavior at the singularities.

For the Bessel case the PI has implemented a partial algorithm with graduate student Ruben Debeerst. To complete it, a number theoretical problem needs to be resolved (see the item on complexity below).

The technical difference between special functions with regular singularities and irregular singularities also implies that an algorithm for the ${}_2F_1$ hypergeometric function will also contain many more cases, and more difficult cases, than irregular singular cases like Bessel. The Bessel case can be completed relatively quickly. The PI estimates that developing a complete algorithm for all cases will take 3 to 5 years.

The current status can be summarized as follows: For differential equations, there is a partial implementation for Bessel type solutions, though the most difficult case remains to be done. The following are in the exploratory stage: essentially all of the theoretical work (see Section 2.2), difference

equations, and almost all of the work (in particular ${}_2F_1$) for differential equations. The explorations and experiments indicate that the project is feasible and that there will be more than enough work for the PI plus at least 2 or 3 graduate students. There is a wide variety of topics for students to work on, ranging from accessible to very challenging.

**Complexity**.
The algorithm should not just be complete in theory, it should also work well in practice, even on complicated examples. This means that steps with high complexity need to be avoided. For example, to complete the Bessel case, the PI encountered a technical problem that could be solved by solving polynomial equations. However, doing so would have lead to a very high complexity, so an alternative was needed. For this technical problem, the polynomial equations will be linearized by computing subfields of an algebraic extension (a topic in the PI's current NSF grant and preprint [53]). This way a polynomial time complexity can be obtained. To obtain an efficient algorithm for the ${}_2F_1$ case, the PI will have to resolve interesting problems with techniques from algebraic number theory and algebraic geometry.

**Summary**.
The main goal in this project is not an improvement of an existing algorithm, or an additional solver in a long list of solvers. Instead, the goal is to develop a complete algorithm for a very important class of problems. Given the enormous range of applications of linear differential and difference equations, the knowledge that complete closed form solvers are actually within reach is a powerful motivation [35] for the PI to work on this topic.

## 1.1 Value of this work to other researchers

Solving recurrence relations (i.e. difference equations) is useful for discovering previously unknown relations between various areas of mathematics. Currently available techniques include [23, 33, 62, 60, 64, 77, 78, 87, 88, 71, 70, 93]. The PI decided to test if his ideas would lead to significant progress in finding such relations. A convenient way to do this was to use Sloane's online database [87]. This database contains many sequences, including thousands that satisfy a linear recurrence. Many of these sequences are known in the literature; the database provides references. What was also very helpful is that the entire set of sequences in the database can be downloaded with one click on a button.

**An experiment**.
Preliminary explorations towards a solver for recurrence relations have been undertaken by Giles Levy, one of the PI's graduate students. The question was: Among those sequences in the database that satisfy a linear second order recurrence, how many are solvable in terms of other sequences in the database under gauge transformations (which involve $+$, $\cdot$, and shifts).

The purpose was to determine experimentally how often such a sequence could be written in terms of other sequences from the literature, using only a subset of the set $O$ from Section 1. That turned out to be the majority. Many new relations were found that were not known to the database, for example,

$$\mathrm{A006605}(n) = \frac{6}{13\,n+9}\left(\mathrm{A027908}(n) + \frac{n+1}{8n+4}\mathrm{A027908}(n+1)\right)$$

a formula that expresses sequence number A006605 in the database (the number of modes of connections of $2n$ points) in terms of A027908 (the coefficient of $x^n$ in $(1+x+x^2)^{2n}$). Thus, the student's current preliminary implementation is already useful to researchers who encounter linear recurrences in their work, because there is already a good chance that this program will discover a relation with other sequences known in the literature. The capabilities of this program will be significantly expanded for Giles' Ph.D work, with the end goal of being provably complete in a large class of closed form sequences.

The PI does not have to search very far to find researchers that will benefit from this project. For example, Philip Bowers in the PI's department encounters sequences and recurrence relations in his computations in quantum mechanics. Closed form solutions of these recurrences are difficult to find by hand, a computer program is needed.

Differential equations are very important for many researchers as well, and a few examples will be given in the remainder of this section. W.N. Everitt sent the PI examples for which he wanted to know closed form solutions. The PI solved Everitt's equations, and was consequently made a co-author of a preprint [29].

From a computer algebra standpoint, this is not how it should be. For example, if the computer returns a polynomial unfactored, then it should not be necessary to ask someone else for a factorization; that polynomial should be irreducible. Nobody ever asks the PI to factor a polynomial; a clear sign that computers do a good job on this task. Likewise, the optimal

situation is that once Everitt asked the computer to solve his equations, it should not have been necessary to ask anywhere else; because if a closed form solution exists, the computer should find it, and if it does not find closed form solutions, then such solutions should provably not exist. This way the arbitrariness is taken out of the process.

It is very useful to have a complete algorithm for a very general notion of solvable in closed form. Then one can no longer encounter solvable examples that the computer leaves unsolved (which at the moment is common [29, 30]). The example at the beginning of Section 1 had five singularities in $\mathbb{C} \bigcup \{\infty\}$. That is more than enough to make it very unlikely to find closed form solutions with current computer algebra systems.

The explanation is as follows. A necessary condition for current algorithms to work is that the set of singularities can be mapped by a Möbius transformation to the set of singularities of a textbook equation. The larger the set of singularities is, the less likely it is that this condition holds.

However, this explanation does not imply that current algorithms are complete for equations with few singularities. To illustrate that, the PI constructed the following example with just one singularity, an irregular singularity at infinity:

$$y'' + (2 - 10x + 4x^2 - 4x^4)y = 0 \tag{1}$$

Current computer algebra systems can find a solution in terms of triconfluent Heun functions, which are not in closed form according to the PI's definition. Finding a solution in terms of Heun functions is better than finding no solution, but closed form solutions are preferable, see the discussion in Section 1 in [67] (by choosing the rigid [59] special functions, the PI's definition of closed form encapsulates precisely those special functions about which a great deal of practically useful information is known, which makes closed form expressions much more manageable in practical computations than other functions such as Heun functions.)

The same equation is also solvable in closed form because the solutions can be written in terms of Airy Ai and Bi functions using field operations, composition, and differentiation:

$$y_1 = (2x^2 + x - 1)\mathrm{Ai}(x^2 - 1) + (2x + 1)\mathrm{Ai}'(x^2 - 1)$$
$$y_2 = (2x^2 + x - 1)\mathrm{Bi}(x^2 - 1) + (2x + 1)\mathrm{Bi}'(x^2 - 1)$$

A complete closed form solver would contain a complete Airy solver and should thus solve equation (1) in closed form.

Converting Heun functions to closed form is an area of research in itself, see for example [21, 28, 57, 67]. This project would provide significant progress for Heun functions. The techniques used thus far for converting Heun functions to closed form are not general enough for examples such as the one above (sums are currently not treated). When this project is completed, closed form expressions can be found automatically whenever they exist.

### 1.2 Value to 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 this. Most of the people who benefit from the PI's work will be unaware of it[1]. In fact, in computer algebra, that is precisely how it should be because 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.

Many branches of science have important impacts on society. Differential and difference equations occur in almost every branch of science, and having closed form solutions is very useful in practical applications for several reasons (see the discussion in Section 1 in [67]). The PI believes that computer algebra is of great value to society, and that within computer algebra, differential and difference equations are among the areas with the highest overall impact. So when a complete closed form solver for such equations is within reach, then that is what the PI should be working on.

## 2 Feasibility and relation to prior work

There are numerous algorithms and tools [1, 3, 14, 15, 16, 17, 18, 19, 22, 23, 24, 33, 39, 43, 44, 50, 52, 61, 62, 63, 68, 72, 77, 78, 79, 83, 84, 89, 90, 92] for solving linear differential and difference equations. Each treats a certain type of solutions. Some of these algorithms (specifically, those that compute exponential and Liouvillian solutions) have all closure properties but no special functions. For other algorithms the reverse is true. However, none of the methods behind these algorithms generalizes to cover all closed form solutions. So the PI will follow a new approach.

[1] The same is also true for the PI's existing algorithms, the PI has met many people at conferences that have used some of the PI's algorithms without being aware of it

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 project would take many years if it were not for the fact that some of the required tools are already available. The most important of these tools are: reduction of order (more details on this will be given in Subsection 2.1 below), software [37] for finding gauge transformations, hypergeometric [23] exponential [22, 74] and Liouvillian [43, 52, 61, 83, 85] solutions, local data at singular points (see [22] resp. [23, 44] for the for the differential resp. difference case), subfields of algebraic extensions [53], etc.

Because of these tools, the PI estimates that complete closed form solvers can be developed in 3 years with some additional time needed for completeness proofs, see Subsection 2.2 below.

### 2.1 Reduction of order

Solving an equation (a differential equation or a recurrence relation) often involves reduction of order, which means writing solutions in terms of solutions of equations of lower order. For instance, a reduction of order is possible if the operator that corresponds to the equation can be factored.

So factoring differential operators is useful for solving differential equations. Likewise, factoring difference operators is useful for solving recurrence relations. The PI has developed algorithms for factoring differential differential operators and for computing first order factors of difference operators (see NSF grants 9805983 and 0098034, and papers [22, 23, 39, 44]). An efficient implementation for higher order factors in the difference case will be available before the starting date of this project, see Section 3.

Factoring is a common way to reduce the order, but not the only way; there exist equations that can be reduced to lower order equations even though the corresponding operator is irreducible. In [81] Michael Singer classifies the ways in which a differential equation can be reduced to equations of order 2. For equations of order 3, there are now efficient implementations available by the PI for all possible cases of reduction to order 2, see [34].

Computing Liouvillian solutions [43, 52, 61, 83, 85] is equivalent to reducing an equation to first order equations (defined over a field extension). Eulerian solutions [81] correspond to a reduction to order 2. Because of these reductions, the primary bottleneck now is to solve equations of order 2 that do not have Liouvillian solutions. So order 2 will have the highest priority, because that will have an immediate impact on higher order equations as well (for more details see [34].)

## 2.2 Mathematical tools needed for this project

To prove that an algorithm for solving differential or difference equations is complete requires structure theorems that describe precisely what type of solutions can actually occur. Here is a structure theorem for the Bessel case.

**Theorem**. Let $K_0$ be the algebraic closure of $\mathbb{C}(x)$. Let $f_1, \ldots, f_n \in K_0$, $\nu_1, \ldots, \nu_n \in \mathbb{C}$ and $r_1, \ldots, r_m \in K_0$. Let $I_\nu$ and $J_\nu$ denote the Bessel I and J functions with parameter $\nu$. Let $K$ be the differential field generated over $K_0$ by the functions $\exp(\int r_i)$, $I_{\nu_i}(f_i)$, $J_{\nu_i}(f_i)$ and their derivatives. Let $L$ be a linear second order differential equation with coefficients in $\mathbb{C}(x)$ and no Liouvillian solutions. If $L$ has a non-zero solution in $K$ then $L$ has a basis of solutions of the form

$$y_1 = (a_1 I_\nu(f) + b_1 I_{\nu+1}(f))\, e^{\int r}, \quad y_2 = (a_2 J_\nu(f) + b_2 J_{\nu+1}(f))\, e^{\int r}$$

where $r, a_1, b_1, a_2, b_2$ and $f^2$ are in $\mathbb{C}(x)$. Moreover, if $L$ is defined over $C(x)$ for some subfield $C \subseteq \mathbb{C}$ then $r, a_1, b_1, a_2, b_2, f^2 \in C(x)$ and $\nu^2 \in C$.

The proof of the above theorem is long but not excessively difficult. However, it still needs to be extended because as stated the Bessel functions are only composed with algebraic functions and not with arbitrary elements of $K$. If a differential equation of order $> 2$ can be solved in terms of Bessel functions then one can use reduction of order [81].

Because of the practical value of the algorithm, the task of designing the algorithm will take priority over the (more difficult) theoretical work of proving completeness. The PI intends to develop complete algorithms within 3 years and completeness proofs within 5 years. The theoretical work is important, because it means that if the algorithm finds no solution, then the user can be confident that there really are no closed form solutions.

The remainder of this subsection will contain technical points. The purpose is to give an indication of the theoretical work that needs to be done, and which tools the PI intends to use.

To extend the theorem to all closure properties, observe that a composition of two Bessel functions will have a double exponential asymptotic behavior and hence such a composition cannot occur as a solution of an equation with rational function coefficients. It is more difficult to prove that when such compositions are combined with other field operations, that the asymptotic behavior will not cancel sufficiently to obtain a solution of an equation with rational function coefficients. To do this, the PI will study the monodromy action (i.e. the effect of analytic continuation). The aim is

to prove that any such cancelation can not persist after applying analytic continuation on paths around the singularities of $L$.

For the ${}_2F_1$ case one depends even more on arguments based on analytic continuation. One way in which the ${}_2F_1$ case differs from the irregular singular cases is that if one composes two ${}_2F_1$'s, then the singularities of the composition do not have an asymptotic growth beyond that what is possible for solutions of equations with rational function coefficients. In fact, a composition of two ${}_2F_1$'s can be a solution of an equation with rational function coefficients (for example, choose their parameters in the Schwartz list [8], so that they are algebraic functions. The composition of two algebraic functions is again algebraic, and every algebraic function satisfies a linear differential equation.)

Suppose $y$ is a composition of two ${}_2F_1$'s neither of which is algebraic. How can one prove (as a step towards a structure theorem) that $y$ can not be a solution of a linear equation $L$ with rational function coefficients? The idea is that $L$ has only finitely many singularities, and that if one applies analytic continuation to $y$ over paths around the singularities of $L$, one will encounter additional singularities of $y$. Then $y$ can not be a solution of $L$.

The deeper mathematical reason that proving structure theorems and designing complete algorithms is feasible is that the selected special functions satisfy rigid differential equations. This means that they are globally determined by their local asymptotic behavior. The PI needs this property to obtain structure theorems. This same property also explains why these functions have so many useful properties.

### 2.3 Can a solver be both complete and efficient?

The PI's approach to find closed form solutions will use a quantitative classification of the asymptotic behavior at the singularities (the so-called generalized exponents [22, 39]. For the analogue in the difference case see [23, 44, 65]). The idea is to try derive enough data about the solutions from their asymptotic behavior so that the remaining data can be found by solving equations. These equations must be linear to ensure the algorithm will be efficient. That can be accomplished more easily for special functions with irregular singularities because the PI's generalized exponents yield more data for irregular singularities than for regular singularities. The example at the beginning of Section 1, which had an irregular singularity at infinity, could be solved [36] in less CPU time than it took to verify the solutions.

For Bessel functions there is a case in which the equations are non-linear

but can be linearized by computing subfields of an algebraic extension. That this works depends on the structure theorem in Section 2.2 (specifically, the fact that $f^2 \in C(x)$ if $L$ is defined over $C(x)$.) This is one of the ways how such theorems help to obtain an efficient algorithm. Such theorems will also be needed to address another efficiency issue, the *field problem* in [23]. The field problem is also the reason that PI has worked on theorems [55] related to reduction of order.

Reduction of order is important to solve equations of order $> 2$. The algorithm in [81] involved solving a system of non-linear equations; a computationally expensive step [25]. The PI developed an algorithm [54] to compute a point on a conic over $C(x)$ that does not involve solving non-linear equations, so that the reduction from order 3 to order 2 can now be done efficiently [34].

As in [22, 23] (item 2b in Section 4), one can combine $p$-curvature techniques with local data to detect rapidly when an equation may have closed form solutions, and if so, which special functions will be involved.

The ${}_2F_1$ case (only regular singularities) will be by far the most involved[2] and will use techniques from algebraic geometry. A large portion of this project is to make sure that solutions will computed efficiently not just in easy cases but in complicated cases as well.

# 3 Current research

The PI is currently working on two projects under the support of the PI's current NSF grant 0511544, whose funds will be depleted by the end of the summer of 2007. The first one is important for this proposal, while the second fits better with the PI's current NSF grant. For reasons explained below, both need to be completed during the summer of 2007, before the requested starting date of this proposal.

**Finish joint project with Barkatou and Bronstein**.
The PI had started a research project on factoring difference operators (see [51]) with Manuel Bronstein before his tragic death. Manuel worked on a similar topic with Moulay Barkatou (systems instead of operators). So it is natural to combine this into a joint three-author paper, and to do this, Moulay Barkatou will visit the PI for one month during the summer of 2007.

[2] Keep in mind that the PI's goal involves many closure properties and that with just one of them, e.g. composition, the problem is already non-trivial, see for example [52, 67, 91].

**Prove new best complexity result for factoring polynomials**.
The second project is the complexity of factoring polynomials with rational coefficients. The PI's algorithm algorithm given in [45] works well in practice but no complexity bound was given in this paper. Then a joint preprint [11] proved a polynomial time complexity bounds for two versions of this algorithm. These bounds did not improve prior complexity bounds. Most unfortunate was that the version [10] with the best performance had the worst complexity bound in our preprint. However, for this version the PI can now prove a complexity bound that actually matches its practical performance. This will be the first improvement in decades in the complexity bound for factoring in $\mathbb{Q}[x]$. The PI plans to work out the details with Andrew Novocin, one of the PI's graduate students. This work must be completed before July when the PI is scheduled to present this at an invited lecture in Edinburgh [38].

# 4 Results from prior NSF support.

The following is an overview of research supported by the PI's prior NSF grants (for details on those grants see Subsection 4.1).

1. Recurrence relations

   (a) In [44] the PI 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 [70]). More recently, this algorithm was developed further in a joint paper [23] with T. Cluzeau. The algorithm is much more efficient than previous algorithms which means that much larger problems can now be handled. An implementation by the PI of this algorithm is available.

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

   (c) A generalization of Gosper's algorithm to $n$'th order recurrences was given in [4]. This algorithm can be used not only to prove but also to find interesting identities.

2. Solving linear differential equations.

   (a) Solving Second Order Linear Differential Equations with Klein's Theorem [52]. The Kovacic algorithm [61] is a famous algorithm

for finding Liouvillian solutions of linear second order differential equations. With the use of Klein's theorem, one can give more compact solutions than those found with the Kovacic algorithm. This makes the solutions more practical. An implementation of this algorithm is available.

(b) A new algorithm for computing exponential solutions was developed in [22]. 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).

(c) An algorithm was developed in [50] for writing solutions of fourth order equations as products of solutions of second order equations.

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

(e) An algorithm for computing Liouvillian solutions of third order differential equations was developed and implemented [43].

3. Factoring polynomials.
The PI developed a new algorithm [45] for factorization of polynomials with rational number coefficients, specifically, the combinatorial problem appearing in Zassenhaus' algorithm is solved efficiently. This algorithm is a significant practical improvement [66]. 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.

4. Evaluating Riemann Theta functions, Riemann matrices.
The following algorithms were developed in [26] as joint work with Bernard Deconinck: monodromy, homology, differentials, and period-matrix (also called Riemann matrix). These algorithms allow to compute numerically in the Jacobian of an algebraic curve, and represent a significant amount of work. This work was followed up in 2004 with a paper [27] on computing Riemann Theta functions. Two implementations (in Maple and in Java) for computing Riemann Theta functions were provided as well. These functions are important for solving differential equations such as the KdV equations, which in turn are important for describing ocean waves (for more see NSF nugget [69]).

5. Descent for differential operators.

The PI has worked on several methods for reducing differential equations to differential equations of lower order. The corresponding differential operator could for example be a product of lower order operators [39]. The PI has also used other constructions (symmetric product [50], symmetric power, or a symmetric power after a gauge transformation) to develop and implement algorithms for reducing the order. In these methods, it is possible that algebraic numbers need to be introduced to perform this reduction in order. To design efficient algorithms, it is important to know in advance which algebraic extensions may occur. The PI has written a joint paper [55] with M. van der Put about the mathematics underlying this issue.

In [81] Singer gave a process for reducing third order linear differential equations to second order equations by finding a point on a conic. In [55] it was shown precisely which conics can occur here. The PI also developed an algorithm [54] to find a point on such a conic, so Singer's reduction to second order can now be performed efficiently, without the need for solving polynomial equations, see [34]. This reduction could be done for recurrence relations as well.

The paper [55] also proves that second order equations are projectively equivalent if and only if the same is true for their symmetric squares. The PI has also developed an algorithm for computing gauge transformations. This algorithm is also useful for computing ladder operators in physics, and is available at [37].

6. Algebraic curves.
   An algorithm was developed in [47] to compute a normal form for hyperelliptic curves. Prior to NSF support, the PI already developed such algorithms for the elliptic case [42] as well as for parametrizing algebraic curves [41] of genus 0. Note that a parametrization is not unique, but only unique up to a Möbius transformation. Consequently, these algorithms often find an answer with much larger coefficients than necessary. An algorithm to address this issue was implemented by the PI's graduate student Andrew Novocin.

7. 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 [48], [49] with M. Monagan on modular algorithms for GCD computation, one for number fields presented with multiple extensions, and one for function fields. These algorithms will

be added to computer algebra systems, so that these systems will handle algebraic extensions significantly faster.

### 4.1 List of prior NSF grants

**Title:** Algorithms for Solving Linear Recurrence Equations.
NSF 9805983, 06/15/98 – 05/31/00, $41,802

**Title:** East Coast Computer Algebra Day 2001.
NSF 0112495, 08/15/01 – 07/31/02, $8,500

**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

**Graduate students:**
NSF grant 0098034 supported graduate student Andrew Novocin as a research assistant. Andrew presented a talk at the ACA'2003 conference and a poster at the ISSAC'2004 conference in Spain. Andrew plans to finish his Ph.D thesis by the end of 2007. A second graduate student, Giles Levy, started working with the PI in 2005. Two more students, Quan Yuan and Cha Yong, will start working with the PI after their qualifying exams in the summer of 2007. Support is requested for the graduate students to offer them valuable opportunities for doing research and to attend conferences.

**Supported conference:**
NSF grant 0112495 supported the ECCAD'01 conference. It also supported travel expenses of participants, mainly of the student participants.

**Implementations:**
The PI's implementations are a valuable contribution to the scientific community and are used by many researchers worldwide. The PI is very grateful for the granted support; without it these implementations could not have been written.

**Papers supported by NSF grants 0098034 and 0511544:**
Journal publications: [45, 5, 22, 27, 55, 2, 23]
Refereed conference publications: [48, 49, 17, 50, 52]
Preprints submitted for publication: [54, 32, 34]
Other preprints: [47, 46, 53, 11, 29]
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. Bronstein, *On Solutions of Linear Functional Systems.* ISSAC'2001 proceedings, 1-6 (2001).

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

[5] 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).

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

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

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

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

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

[11] K. Belabas, J. Klüners, M. van Hoeij, and A. Steel *Factoring polynomials over global fields*, preprint.

[12] 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

[13] M. Bronstein *Symbolic Integration Tutorial*, (1998). http://www-sop.inria.fr/cafe/Manuel.Bronstein/publications/issac98.pdf

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

[15] M. Bronstein, *Linear Ordinary Differential Equations: breaking through the order 2 barrier*, Proceedings of ISSAC'92, 42-48, (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, *Computing Mathieu function solutions for linear ODEs*, http://www.scg.uwaterloo.ca/~ecterrab/odetools/mathieu_function_solutions.html

[20] E.S. Cheb-Terrab, *The Function Advisor project: an alive Computer Algebra Handbook of Special Functions*, Proceedings of the Maple Summer Workshop - Waterloo, (2002).

[21] 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).

[22] 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).

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

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

[25] D. Cox, J. Little, and D. O'Shea, *Ideals, varieties, and Algorithms* Undergraduate Texts in Math, Springer (1992).

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

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

[28] H. Exton, *A new solution of the biconfluent Heun equation*, Rendiconti di Mathematica Serie VII, **18**, 615-622 (1998).

[29] 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`

[30] 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).

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

[32] A. Galligo and M. van Hoeij, *A Geometric Approach to Factoring Bivariate Approximate Polynomials*, submitted to ISSAC'2007.

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

[34] M. van Hoeij, *Solving Third Order Linear Differential Equations in Terms of Second Order Equations*, submitted to ISSAC'2007, `www.math.fsu.edu/~hoeij/files/ReduceOrder/Paper`

[35] M. van Hoeij, *Why I do what I do*. `www.math.fsu.edu/~hoeij/why`

[36] *Solution to example 1*, `www.math.fsu.edu/~hoeij/files/ODE3`

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

[38] Journées Arithmétiques, conference in Edinburgh, July 2 – 6 (2007). `http://icms.org.uk/event.php?id=43`

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

[40] 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).

[41] M. van Hoeij, *Rational Parametrizations of Algebraic Curves using a Canonical Divisor*, J. Symbolic Computation, **23**, 209-227 (1997).

[42] M. van Hoeij, *An algorithm for computing the Weierstrass normal form*, ISSAC '95 Proceedings, 90-95 (1995).

[43] 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). Implementation: `http://www.math.fsu.edu/~hoeij/files/impr_order3` (2004).

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

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

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

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

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

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

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

[51] M. van Hoeij, *Factorization and hypergeometric solutions of linear recurrence systems*, slides for Manuel Bronstein's conference, `www.math.fsu.edu/~hoeij/papers.html` (2006).

[52] M. van Hoeij and J.A. Weil, *Solving Second Order Linear Differential Equations with Klein's Theorem*, ISSAC'05 Proceedings, 340-347, (2005).

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

[54] M. van Hoeij, J. Cremona, *Solving conics over function fields*, accepted for publication in JNTB.

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

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

[57] A.M. Ishkhanyan and K.-A. Suominen, *New solutions of Heun's general equation*, J. Phys. A: Math. Gen. **36**, (2003)

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

[59] Nicholas M. Katz, *Rigid Local Systems*, Princeton University Press, (1995).

[60] W. Koepf, *hypergeometric summation package*, `http://www.mathematik.uni-kassel.de/~koepf/Publikationen`

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

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

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

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

[65] A.H.M. Levelt, A. Fahim. *Characteristic classes for difference operators.* Compos. Math. **117**, No.2, 223-241 (1999).

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

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

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

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

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

[71] M. Petkovšek, H. Wilf and D. Zeilberger, $A{=}B$, available for download at `http://www.cis.upenn.edu/~wilf/AeqB.html`

[72] 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).

[73] M. van der Put, *Galois Theory of Differential Equations, Algebraic Groups and Lie Algebras*, Journal of symbolic computation **28**, 441-472 (1999).

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

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

[76] R. H. Risch, *The Problem of Integration in Finite Terms*, Transactions of the American Mathematical Society, **139**, 167-189 (1969).

[77] B. Salvy and P. Zimmermann, *GFUN: A Maple Package for the Manipulation of Generating and Holonomic Functions in One Variable*, ACM Transactions on Mathematical Software, **20**, (1994).

[78] C. Schneider, *The Summation Package Sigma*, http://www.risc.uni-linz.ac.at/people/cschneid

[79] 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).

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

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

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

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

[84] 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).

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

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

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

[88] N. Sloane, *The Email Servers and Superseeker*, `http://www.research.att.com/~njas/sequences/ol.html`

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

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

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

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

[93] D. Zeilberger, *The Method of Creative Telescoping.* J. Symbolic Computation, **11**, 195-204 (1991).