Algorithms for Solving Linear Ordinary Differential Equations

Winfried Fakler
IAKS, Universität Karlsruhe, fakler@ira.uka.de

This article presents the new domain of linear ordinary differential operators and shows how it works in a few examples. Furthermore, in a very informal way the algebraic point of view dealing with ordinary differential equations will be introduced. Using such tools allows to develop general algorithms for solving linear equations.

Introduction

In designing algorithms for solving differential equations one necessarily has to work at the (algebraic) structure of these equations. On one hand this leads to a classification of special types of differential equations. But it never can be complete. On the other hand this will give a classification like linear and nonlinear, ordinary and partial differential equations.

A further decision is, what kind of solutions one would like to determine. For example, one can search for power series solutions, formal or closed-form (symbolic) solutions. This article treats symbolically constructable solutions of linear ordinary differential equations.

Let

$$L(y) = y^{(n)} + a_{n-1}y^{(n-1)} + \ \dots \ + a_1 y' + a_0 y = 0 \qquad (a_i \in k)$$

be a nth order homogeneous linear differential equation over the coefficient field k, where k is e.g. the field of rational functions. It is well-known, that a nth order linear differential equation has exactly n linearly independent solutions which form a vector space.

Constructable solutions can be grouped together to classes of solutions.

Rational solutions.
Rational solutions are functions lying in k. For that class J. Liouville already gave an algorithm in 1833, but only when k is the rational function field. More general versions are presented by Singer (1991) and Bronstein (1992).

Algebraic solutions.
Algebraic solutions are functions, which lie in an algebraic extension of k, i.e. they satisfy an irreducible polynomial over k. An example for this class is

$$\sqrt[3]{1 - \sqrt{x}}.$$

Many renowned mathematicians like Pépin, Fuchs, Klein, Jordan were searching for an algorithm for algebraic solutions. Today, there exists an algorithm from Singer, but it is far from being satisfactory.

Liouvillian solutions.

Liouvillian solutions are functions, which can be constructed from the rational functions by successively substituting nested algebraic functions, integrals and exponential of integrals. An example for such a construction is

$$x \xrightarrow{\sqrt{\ }} \sqrt{x} \xrightarrow{e^{\int}} \exp\left[\int \sqrt{x}\right].$$

This class of functions includes also functions like the trigonometric functions and logarithms. In principle it consists of nearly all closed-form functions. An important exception are the Bessel functions (and other special functions).

Exponential solutions.

Exponential solutions are functions, whose logarithmic derivatives lie in k. If y is a solution, then y'/y is its logarithmic derivative. An example is

$$y = \exp(x^3),$$

since $y'/y = 3x^2$ lies in k. For this article, exponential functions forms the most important subclass of liouvillian functions. Without an algorithm for them, it is not possible to give an algorithm for liouvillian solutions. Fortunately, there are algorithms for exponential solutions. The very first one stems from Beke in 1894.

Dividing functions into one of these classes is not always unique, e.g. for the function $y = \sqrt{x}$, it is possible to attach it either to the algebraic functions or to the exponential, since $y'/y = 1/2x \in k$. Furthermore, it can be hard to decide for a given function whether it is liouvillian and how one could find the simplest construction.

To every linear ordinary differential equation

$$L(y) = y^{(n)} + a_{n-1}y^{(n-1)} + \ \ldots\ + a_1 y' + a_0 y = 0$$

one can associate a linear operator

$$L(D)[y] = (D^n + a_{n-1}D^{n-1} + \ \ldots\ + a_1 D + a_0)[y] = 0$$

in a unique way. Here, D^i is only another notation for the i-th derivation of y. One calls such an operator a linear ordinary differential operator. The mathematical structure of linear ordinary differential operators is a ring. This structure and its conversion in *MuPAD* is described in the next section.

The LODO Domain Constructor

The ring of linear ordinary differential operators $k[D]$ is presented in *MuPAD* as the domain constructor `Linear-OrdinaryDifferentialOperator`. From the mathematical point of view linear differential operators generate a left skew polynomial ring of derivation type. The elements of such a ring are called skew polynomials or Ore polynomials. For Ore polynomials the usual polynomial addition holds. Only the multiplication is different. It is declared as an extension of the rule for $a \in k$

$$Da = aD + a'$$

to arbitrary Ore polynomials. Multiplication of Ore polynomials is in fact operator composition. Therefore, $k[D]$ is not a commutative ring. This means, there is a left and right division. Indeed, there exists an extended Euclidean algorithm and it is possible to determine for any two nontrivial elements a smallest nontrivial common left multiple. This is the so-called *Ore condition*, i.e. skew polynomials are left Ore rings. Linear ordinary differential operators are even left and right Ore rings.

Starting from this mathematical background the category constructor `UnivariateSkewPolynomialCat` was generated, in which already all operations for univariate skew polynomials independent of the representation are implemented. For a true representation the following domains hierachy was created:

$$\begin{array}{c}\texttt{BaseDomain}\\ \downarrow\\ \texttt{PolynomialExplicit}\\ \downarrow\\ \texttt{UnivariatePolynomial}\\ \downarrow\\ \texttt{UnivariateSkewPolynomial}\\ \downarrow\\ \texttt{LinearOrdinaryDifferentialOperator}\end{array}$$

In this way, based on polynomials it was possible to implement the new domains in a reasonable time. Simply the polynomial multiplication needs to be overloaded by the new noncommutative multiplication and all the resulting left and right operations had to be implemented. Altogether it is an example for the advantages of the domains constructor concept. Based on the domain `UnivariateSkewPolynomial`, it would be possible to generate beside the domain `LinearOrdinaryDifferentialOperator` e.g. a domain for linear ordinary difference operators without need for implementing all operations once more.

To create a LODO domain one has to choose a variable for the operator, e.g. 'Df', a variable with respect to which one wants to differentiate, and optionally a coefficient field or ring of characteristic zero from the domains package. Note, in *MuPAD* 'D' is not a variable, since it is predefined as an operator. A LODO domain can be created for example with the call

```
>> EF := Dom::ExpressionField(normal):
>> lodo := Dom::LinearOrdinaryDifferentialOperator(Df,x,EF);
```

There are more possiblities for generating differential operators, e.g. one can generate the same operator by a vector

```
>> lodo([x+1, sin(x), 1]);
```

or by a differential equation

```
>> lodo(diff(y(x),x,x)+sin(x)*diff(y(x),x)+(x + 1)*y(x),y(x));
```

and of course by an operator polynomial

```
>> A:=lodo(Df^2+sin(x)*Df+x+1);

                     2
                   Df  + sin(x) Df + (x + 1)

>> B:=lodo(Df+x);

                            Df + x
```

The product of the two operators one gets with

```
>> P:=A*B;

   3                   2                                            2
 Df  + (x + sin(x)) Df  + (x + x sin(x) + 3) Df + (x + sin(x) + x )
```

That this product is really an operator composition one can test with

```
>> expand((A*B)(y(x),Unsimplified) - A(B(y(x),Unsimplified),Unsimplified));

                                       0
```

One can also see from

```
>> lodo(Df)*lodo(x), lodo(x)*lodo(Df);

                               x Df + 1, x Df
```

that the defined multiplication satisfies the rule above and that it is noncommutative. The option 'Unsimplified' is necessary, since the function `func_call` of the LODO domain is primarily intended for the zero test of solutions. Naturally it is possible to evaluate operators, but there are internal manipulations which leave the solution space unchanged, however it will change the resulting expression. By the given option this will be suppressed. It is also possible for example to compute a right division.

```
>> t:=P::rightDivide(P,A);

             table(
               remainder = (- cos(x) + 2 ) Df + (sin(x) - 1),
               quotient = Df + x
             )


>> t[quotient]*A+t[remainder];

      3                  2                                          2
    Df  + (x + sin(x)) Df  + (x + x sin(x) + 3) Df + (x + sin(x) + x )
```

In short, a LODO domain contains among other things the operations left/right{Divide, Quotient, Remainder, Gcd, Lcm, ExtendedEuclid} and allows to determine the adjoint operator,

```
>> P::adjoint(P);

    3                  2
- Df  + (x + sin(x)) Df  + (- x + 2 cos(x) - x sin(x) - 1 ) Df +

                                  2
(x - sin(x) - x cos(x) + x  - 1)
```

symmetric powers (`symmetricPower`) of an operator and currently limited factoring and computing zeros or solutions of operators with rational function coefficients. For demonstrating that, here an example:

```
>> L:=lodo(Df^4+(2*x-1)/(2*x*(x-1))*Df^3+(143*x-147)/(784*x^2*(x-1))*Df^2\
&>   +(-18*x+21)/(32*x^3*(x-1))*Df+(2349*x-2940)/(3136*x^4*(x-1)));
```

```
  4   /     2 x - 1     \   3   /     143 x - 147      \   2
Df  + | ------------- | Df  + | ------------------ | Df  +
      |             2 |       |       2        3   |
      \ - 2 x + 2 x   /       \ - 784 x  + 784 x   /

/   - 18 x + 21    \          2349 x - 2940
| ---------------- | Df + --------------------
|       3       4  |             4         5
\ - 32 x  + 32 x   /      - 3136 x  + 3136 x
```

```
>> Factor(L);
```

```
/   2   /     2 x - 1     \               1          \ /       1  \
| Df  + | ------------- | Df - ----------------- | | Df + --- |
|       |             2 |                        2  | \      4 x /
\       \ - 2 x + 2 x   /      - 196 x + 196 x   /

/       1  \
| Df - --- |
\      4 x /
```

An operator which decomposes into factors is called *reducible*, if it is not reducible, it is called *irreducible*. Currently, the function `Factor` can only find left and right factors of degree 1, which means that a decomposition in irreducible factors is guaranteed only for operators up to third degree. Nevertheless it is possible to find decompositions of higer degree operators. The situation for computing liouvillian zeros is quite similar. Finding *all* liouvillian zeros can currently only be guaranteed for second degree operators and reducible operators of third degree. But one can also find liouvillian zeros of higher degree operators.

```
>> sols:=L::liouvillianZeros(L):
>> map(sols,combine@simplify@expand@eval);
```

```
{  1/4      3/4      3/4     /  1/4     / acosh(2 x - 1) \     \
{ x   , 2 x   , 2 x    int| x    exp| -------------- |, x | -
{                            \          \       14       /     /

      1/4     /  3/4     / acosh(2 x - 1) \     \
   2 x    int| x    exp| -------------- |, x |,
              \          \       14       /     /

      3/4     /  1/4     /   acosh(2 x - 1) \     \
   2 x    int| x    exp| - -------------- |, x | -
              \          \         14       /     /

      1/4     /  3/4     /   acosh(2 x - 1) \     \ }
   2 x    int| x    exp| - -------------- |, x | }
              \          \         14       /     / }
```

```
>> map(%,L);
```

```
{0}
```

The last call shows that the determined functions are in fact zeros of the operator.

For readers who want to know more about Ore polynomials and linear ordinary differential operators we refer to Bronstein and Petkovšek [2] and Ore [6]. The currently most efficient method for factoring differential operators is described in van Hoeij [11]. Information about the domains package one can find in the online help system of *MuPAD*.

An Algebraic Algorithm

Algorithms computing liouvillian solutions of second order linear differential equations have been developed at the end of the last century. The first complete and implemented algorithm for second order equations stems from Kovacic in 1977, see [5]. A considerably simplification and much more efficient variant of this algorithm was recently presented from Ulmer and Weil [10] and already in 1981 Singer gave an algorithm computing liouvillian solutions for equations of arbitrary order. Unfortunately this algorithm has such an enormous complexity, that it is even not implemented for second order equations. A new development by Singer and Ulmer [8] could close this gap.

All modern solution procedures are in principle based on differential Galois theory. Because of the vector space structure of a solution set in this theory one associate a linear group to every linear differential equation. From this group, it is now possible to draw conclusions about the solutions. For example it is known, that if a linear differential equation has a liouvillian solution y, then it has also a liouvillian solution of which the logarithmic derivative y'/y is algebraic of bounded degree. On the basis of this Singer theorem, one can reduce the problem of finding liouvillian solutions to the problem of finding a minimal polynomial, thus an algebraic equation. Hence, it is an algebraic problem. The considerably higher efficiency of the Ulmer-Weil algorithm compared with Kovacic's algorithm comes from the different calculation of the coefficients of the minimal polynomial. In the Ulmer-Weil algorithm they are simply determined via recursion.
Even, when the Galois group of an irreducible differential equation is finite, one can determine a minimal polynomial of a solution. But for this the Galois group must be explicitely known. For finite primitive groups, L. Fuchs determined in the years from 1875 to 1878 such a method for second order equations. In Singer und Ulmer [7] this method was extended to higher order equations. Minimal polynomials for all finite groups are given in Fakler [3].

For the reader who wants to know more about differential Galois theory we refer to Kaplansky [4] and to the references in the given articles. In figure 1 one can find an overview about the methods for second order differential equations.

In the following we outline the method given in Fakler [3] for second order differential equations and we show how one can use it together with a factorization to compute liouvillian solutions for many higher order equations.

Let

$$L(y) = y'' + a_1 y' + a_0 y = 0, \quad a_i \in k(x)$$

be a differential equation with rational function coefficients over a field k of charakteristic 0. This will be satisfied e.g. for $k = \mathbb{Q}$. Further, let $\{y_1, y_2\}$ be a fundamental set of solutions for $L(y) = 0$ and $\mathcal{G}(L)$ its Galois group. One can imagine it as a (2×2) matrix group, i.e. $\mathcal{G}(L)$ is a subgroup of the general linear group GL$(2,\ k)$.

For using the powerful tools from Galois theory one has to take care that the Galois group of the equation is unimodular, which means that the determinants of all matrices are 1. Then the Galois group is a subgroup of the special linear group SL$(2,\ k)$. This is necessary, since only these groups are all known. However, it is not a problem because e.g. with the transformation $y = z \cdot \exp(-\frac{\int a_1}{2})$ we can guarantee it. For that reason, we assume from now on that the Galois group is unimodular.

One distinguishes, if the Galois group is reducible or imprimitive (which means here, that non-zero elements are only in the main diagonal or only outside the main diagonal) or if it is isomorph to the tetrahedral or octahedral or icosahedral group.

$$L(y) = y'' + a_1 y' + a_0 y = 0, \quad a_i \in k(x)$$

$$\Downarrow$$

Differential Galois theory
$\rightsquigarrow \mathcal{G}(L) \subseteq \mathrm{SL}(2,\, k)$
p_i's invariant under $\mathcal{G}(L)$

$$\Downarrow$$

- Kovacic, Ulmer-Weil: $\mathbf{U} = \frac{y'}{y}$
  $\bar{P}(\mathbf{U}) = \mathbf{U}^m + b_{m-1}\mathbf{U}^{m-1} + \ldots + b_0 = 0$
- Fuchs, Singer-Ulmer:
  $P(\mathbf{Y}) = \mathbf{Y}^{d \cdot m} + p_{m-1}\mathbf{Y}^{d\cdot(m-1)} + \ldots + p_0 = 0$
- Fuchs, Fakler: solutions by formulas

Figure 1: Principle of algebraic algorithms.

That the Galois group of a second order equation is reducible is equivalent to the existence of an exponential solution.

For distinguishing the remaining cases the so-called symmetric powers of $L(y) = 0$ are used. Symmetric powers $L^{\circledS m}(y) = 0$ are from $L(y) = 0$ constructable differential equations with the property that all mth power products of solutions of $L(y) = 0$ are solutions of it. The second symmetric power $L^{\circledS 2}(y) = 0$ e.g. has the solution space $\{y_1^2, y_1 y_2, y_2^2\}$. Rational solutions of symmetric powers $L^{\circledS m}(y) = 0$ correspond to homogeneous (polynomial) invariants of degree m of $\mathcal{G}(L)$. Since the invariants of all the Galois groups are known, it is possible to distinguish the following cases for irreducible second order equations: If $L^{\circledS m}(y) = 0$ has a nontrivial rational solution for

- $m = 4$: $\mathcal{G}(L)$ is imprimitive
- $m = 6$, $(m \neq 4)$: $\mathcal{G}(L) \cong$ tetrahedral group
- $m = 8$, $(m \neq 4, 6)$: $\mathcal{G}(L) \cong$ octahedral group
- $m = 12$, $(m \neq 4, 6, 8)$: $\mathcal{G}(L) \cong$ icosahedral group
- $(m \neq 4, 6, 8, 12)$: $\mathcal{G}(L) \cong \mathrm{SL}(2, k)$.

More, one gets from that a beautiful criterion for when an irreducible second order differential equation with unimodular Galois group has liouvillian solutions namely if and only if its twelveth symmetric power has a nontrivial rational solution.

In the imprimitive case the fourth symmetric power $L^{\circledS 4}(y) = 0$ has a rational solution r (in one case, there are two rational solutions, see figure 2). If W denotes the Wronskian of $L(y) = 0$, which is, by the way, computable from $W = \exp(\int a_1)$, then

$$y_1 = \sqrt[4]{r}\, e^{-\frac{C}{2}\int \frac{W}{\sqrt{r}}} \quad \text{and} \quad y_2 = \sqrt[4]{r}\, e^{\frac{C}{2}\int \frac{W}{\sqrt{r}}}$$

are the desired solutions. The remaining constant C can be computed with the determining equation

$$\frac{4r''r - 3(r')^2}{16r^2} + \frac{W^2}{4r}C^2 - \frac{r'}{4r}a_1 + a_0 = 0.$$

Unfortunately, for the rest of the cases there are no direct solution formulae. However, it remains possible to compute the minimal polynomial of a solution via solution formulae. One recalls, in the Ulmer-Weil algorithm the coefficients of the minimal polynomial of the logarithmic derivative of a solution are determined by recursion. Here the coefficients are computed by determining a constant. With the tetrahedral group we demonstrate how this procedure will work. For the octahedral and icosahedral group the process is similar.

In a precomputation one determines a minimal polynomial decomposed into invariants for the tetrahedral group:

$$\mathbf{Y}^{24} + 10I_2\mathbf{Y}^{16} + 5I_3\mathbf{Y}^{12} - 15I_2^2\mathbf{Y}^8 - I_2I_3\mathbf{Y}^4 + I_1^4.$$

To the invariant $\bar{I}_1 = 4I_1$ corresponds a rational solution r of the sixth symmetric power $L^{ⓢ 6}(y) = 0$. Since solutions of differential equations can only be unique up to a constant, one has to determine a constant c such that $\bar{I}_1 = c \cdot r$. To do this, one can use the syzygy between the invariants from which one gets the determining equation

$$\left(25J(r,H(r))^2 + 64H(r)^3\right)c^2 + 10^6 \cdot 108r^4 = 0,$$

where

$$H(f) = \frac{m-1}{W^2}\left[\left(\frac{f'}{f}\right)^2 + m\left(\frac{f'}{f}\right)' + ma_1\left(\frac{f'}{f}\right) + m^2a_0\right]f^2$$

is the Hessian – here with $m = 6$ – and

$$J(f,g) = \frac{mfg' - nf'g}{W}$$

the Jacobian with $m = 6$ and $n = 8$. The invariant I_2 one then gets from the Hessian ($m = 6$) by

$$I_2 = \frac{1}{400} \cdot c^2 \cdot H(r),$$

while invariant I_3 can be computed with the Jacobian by ($m = 6$, $n = 8$)

$$I_3 = \frac{1}{3200} \cdot c^3 \cdot J(r, H(r)).$$

In figure 2 is a summary of the just represented method. As an application, we now solve the famous example of Kovacic.

Example.
The second order differential equation

$$L(y) = y'' + \left(\frac{3}{16x^2} + \frac{2}{9(x-1)^2} - \frac{3}{16x(x-1)}\right)y = 0$$

possess no exponential solution and its fourth symmetric power $L^{ⓢ 4}(y) = 0$ has no nontrivial rational solution. But then $L^{ⓢ 6}(y) = 0$ possesses the rational solution $r = x^2(x-1)^2$. This means that the Galois group $\mathcal{G}(L)$ is isomorphic to the tetrahedral group.
For $W = 1$, the Hessian and Jacobian are computed with

$$H(r) = \frac{25}{4}x^2(x-1)^3 \qquad \text{and} \qquad J(r,H(r)) = -\frac{25}{2}x^3(x-1)^4(x-2).$$

From that, one gets the determining equation

$$\begin{aligned}(c^2+27648)x^{16} + (-8c^2-221184)x^{15} + (28c^2+774144)x^{14} + \\ (-56c^2-1548288)x^{13} + (70c^2+1935360)x^{12} + (-56c^2-1548288)x^{11} + \\ (28c^2+774144)x^{10} + (-8c^2-221184)x^9 + (c^2+27648)x^8 &= 0,\end{aligned}$$

Input: $L(y) = 0$ with $\mathcal{G}(L) \subseteq \mathrm{SL}(2, \bar{\mathbb{Q}})$
Output: set of liouvillian solutions or minimal polynomial of a solution

1. $L(y) = 0$ is reducible?
   compute an exponential and a liouvillian solution

2. $L^{Ⓢ 4}(y) = 0$ has a nontrivial rational solution r?
   (a) only one? $y_{1,2} = \sqrt[4]{r}\,\exp\left[\pm\frac{C}{2}\int\frac{W}{\sqrt{r}}\right]$
   (b) two? $r := c_1 r_1 + c_2 r_2$, then (a)

3. $L^{Ⓢ m}(y) = 0$, $m \in \{6, 8, 12\}$ has a nontrivial rational solution r?

   - $m = 6$
     $(25J(r, H(r))^2 + 64H(r)^3)\,c^2 + 10^6 \cdot 108r^4 = 0 \quad (n = 8)$
     $I_1 = \frac{1}{4} \cdot c \cdot r, \quad I_2 = \frac{1}{400} \cdot c^2 \cdot H(r), \quad I_3 = \frac{1}{3200} \cdot c^3 \cdot J(r, H(r))$
     $P(\mathbf{Y}) = \mathbf{Y}^{24} + 10I_2\mathbf{Y}^{16} + 5I_3\mathbf{Y}^{12} - 15I_2^2\mathbf{Y}^8 - I_2 I_3\mathbf{Y}^4 + I_1^4$
   - $m = 8$
     $(49J(r, H(r))^2 + 144H(r)^3)\,c - 118013952r^3H(r) = 0 \quad (n = 12)$
     $I_1 = -\frac{1}{16} \cdot c \cdot r, \quad I_2 = \frac{1}{150528} \cdot c^2 \cdot H(r)$
     $P(\mathbf{Y}) = \mathbf{Y}^{48} + 20I_1\mathbf{Y}^{40} + 70{I_1}^2\mathbf{Y}^{32} + (2702{I_2}^2 + 100{I_1}^3)\mathbf{Y}^{24} +$
     $(-1060I_1{I_2}^2 + 65{I_1}^4)\mathbf{Y}^{16} + (78{I_1}^2{I_2}^2 + 16{I_1}^5)\mathbf{Y}^8 + {I_2}^4$
   - $m = 12$
     $(121J(r, H(r))^2 + 400H(r)^3)\,c + 708624400 \cdot 1728r^5 = 0 \quad (n = 20)$
     $I_1 = \frac{1}{125} \cdot c \cdot r, \quad I_2 = \frac{1}{121 \cdot 34375} \cdot c^2 \cdot H(r), \quad I_3 = \frac{11}{2420 \cdot 3125} \cdot c^3 \cdot J(r, H(r))$
     $P(\mathbf{Y}) = \mathbf{Y}^{120} + 20570I_2\mathbf{Y}^{100} + 91I_3\mathbf{Y}^{90} - 86135665{I_2}^2\mathbf{Y}^{80} - 78254I_2I_3\mathbf{Y}^{70} +$
     $(14993701690{I_2}^3 + 11137761250{I_1}^5)\mathbf{Y}^{60} + 897941{I_2}^2I_3\mathbf{Y}^{50} +$
     $(-11602919295{I_2}^4 + 273542733750{I_1}^5I_2)\mathbf{Y}^{40} + (-151734{I_2}^3 - 6953000{I_1}^5)I_3\mathbf{Y}^{30} +$
     $(503123324{I_2}^5 - 7854563750{I_1}^5{I_2}^2)\mathbf{Y}^{20} + (1331{I_2}^4 + 500{I_1}^5I_2)I_3\mathbf{Y}^{10} + 3125{I_1}^{10}$

4. $L(y) = 0$ has no liouvillian solution.

Figure 2: Sketch of the second order algorithm.

respectively e.g. for the regular point $x_0 = 2$ the equation

$$c^2 + 27648 = 0.$$

Hence, $c = \pm 96\sqrt{-3}$. Substituting

$$\begin{aligned} I_1 &= \frac{1}{4} \cdot c \cdot r = 24\sqrt{-3}\, x^2(x-1)^2, \\ I_2 &= \frac{1}{400} \cdot c^2 H(r) = -432x^2(x-1)^3, \\ I_3 &= \frac{1}{3200} \cdot c^3 J(r, H(r)) = 10368\sqrt{-3}\, x^3(x-1)^4(x-2) \end{aligned}$$

in the minimal polynomial decomposed into invariants, we obtain the desired minimal polynomial of a solution:

$$\begin{aligned} P(\mathbf{Y}) = \mathbf{Y}^{24} &- 4320x^2(x-1)^3\mathbf{Y}^{16} + 51840\sqrt{-3}\, x^3(x-1)^4(x-2)\mathbf{Y}^{12} - 2799360x^4(x-1)^6\mathbf{Y}^8 \\ &+ 4478976\sqrt{-3}\, x^5(x-1)^7(x-2)\mathbf{Y}^4 + 2985984x^8(x-1)^8. \end{aligned}$$ ◇

Solutions of the minimal polynomial $P(\mathbf{Y}) = 0$ are also solutions of the differential equation $L(y) = 0$. From the Galois theory one knows, that for solvable groups the solutions can be represented in radicals (nested root expressions). Therefore, in the cases of the tetrahedral or octahedral group it would be possible to compute radical expressions of the solutions. But such a calculation has an enormous complexity and there is no implementation in any computer algebra system for performing this operation known to the author. However, in the icosahedral case it is only possible to represent a solution implicit as solution of a minimal polynomial.

With this algorithm one can even find solutions of higher order reducible equations. For this, one has to factorize the associated operator and beginning from the right to solve the factors. With the method of variation of parameters which was introduced by Lagrange one can construct solutions of the equation from solutions of the factors. An example for that is already given in the last section. Unfortunately, decomposing into factors is not unique. To determine in this way as many as possible (all) liouvillian solutions one can compute from a given factorization with the algorithm from Tsarev [9] all the other possible factorizations. By a complete implementation of factoring and e.g. by an implementation of the algorithm from Singer and Ulmer [8], one then could find all liouvillian solutions.

I thank Frank Postel for his invitation to present my implementations and results from [3] here, Eckhard Pflügel for implementing the algorithm for exponential solutions, Paul Zimmermann for including my algorithms into the ODE solver and for many helpful discussions, Ralf Hillebrand for including some of my special wishes into the *MuPAD* system and for his help in solving problems and last but not least Werner Seiler for proofreading this article.

References

[1] Barkatou, M. (1997). *An Efficient Algorithm for Computing Rational Solutions of Systems of Linear Differential Equations*. Preprint.

[2] Bronstein, M., Petkovšek, M. (1996). *An introduction to pseudo-linear algebra*. Theor. Comp. Science **157**, No. 1.

[3] Fakler, W. (1997). *On second order homogeneous linear differential equations with Liouvillian solutions.* Theor. Comp. Science **187** (1-2), 27-48.

[4] Kaplansky, I. (1957). *Introduction to differential algebra.* Paris: Hermann.

[5] Kovacic, J. (1986). *An algorithm for solving second order linear homogeneous differential equations.* J. Symb. Comp. **2**, 3-43.

[6] Ore, O. (1933). *Theory of non-commutative polynomials.* Ann. of Math. **34**, 480-508.

[7] Singer, M.F., Ulmer, F. (1993). *Liouvillian and Algebraic Solutions of Second and Third Order Linear Differential Equations.* J. Symb. Comp. **16**, 37-73.

[8] Singer, M.F., Ulmer, F. (1996). *Linear Differential Equations and Products of Linear Forms.*. Preprint. To appear in J. Pure and Applied Algebra.

[9] Tsarev, S.P. (1996). *An Algorithm for Complete Enumeration of All Factorizations of a Linear Ordinary Differential Operator*. In: Proceedings of ISSAC'96, 226-231.

[10] Ulmer, F., Weil, J.A. (1996). *Note on Kovacic's Algorithm.* J. Symb. Comp. **22**, 179-200.

[11] van Hoeij, M. (1996). *Factorization of Linear Differential Operators*. PhD thesis, University of Nijemegen.

[12] Zwillinger, D. (1992). *Handbook of differential equations*. 2nd ed. San Diego: Academic Press.