A DISCUSSION AND IMPLEMENTATION OF KOVACIC'S ALGORITHM FOR ORDINARY DIFFERENTIAL EQUATIONS

by

Carolyn J. Smith

Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
N2L 3G1

Research Report CS-84-35
October 1984

A Discussion and Implementation of Kovacic's Algorithm for Ordinary Differential Equations

by

Carolyn J. Smith

Table of Contents

Index of Definitions

Preface

This essay was written in partial fulfillment of the requirements for the degree of Master of Mathematics in Computer Science at the University of Waterloo.

Thanks go to all the members of the Maple group for their assistance during the implementation of the algorithm.

Special thanks go to my supervisor, Bruce Char, for his support and assistance through the duration of this work.

CHAPTER 1

Introduction

1.1. Overview

The following essay is a discussion of an algorithm by J. Kovacic [3] to solve certain second order ordinary differential equations, and of an implementation of this algorithm in Maple.

In the remainder of this chapter, the exact form of the problem to be solved is given. In chapter 2, some of the theory that is intrinsic to the algorithm is developed and the general form of the algorithm is specified.

In chapter 3, Kovacic's algorithm is presented and proved correct. In chapter 4, a variant of the algorithm developed by B. D. Saunders for implementation in Macsyma [5] is discussed and some corrections made. The corrected algorithm is also verified to correspond to Kovacic's algorithm.

In chapter 5, the implementation of the algorithm is discussed and some problems with Maple as the implementation system are mentioned. Some recommendations for future work on Maple and on the implementation of the algorithm are made. Source code is presented in appendix A. Some examples of the use of the algorithm and of the implementation are presented in appendix B.

1.2. Purpose

The equation to be solved is assumed of the following form:

$$a \cdot z'' + b \cdot z' + c \cdot z = 0 \qquad (1.2a)$$

where a, b and c are rational functions of a (complex) variable x with coefficients in the field of complex numbers $\mathbf{C}$, $a \neq 0$ and z is a (complex) function of x.

The goal of the algorithm is to find one Liouvillian solution of the equation. A solution is Liouvillian if it is an element of a Liouvillian field, where a Liouvillian field is defined as follows:

Definition: Let F be a differential field of functions of a complex variable x, that contains $\mathbf{C}(x)$, i.e. F is a field of characteristic zero with a differentiation operator $'$ with the following two properties: $(a+b)' = a'+b'$ and $(ab)' = ab'+a'b$, for all a and b in F. (The characteristic of a field is the least integer $q > 0$ for which $qa = 0$ for all a in the field. If no such q exists, then the characteristic of the field is zero.) F is Liouvillian if there exists a tower of differential fields

$$\mathbf{C}(x) = F_0 \subseteq F_1 \subseteq \cdots \subseteq F_n = F$$

such that for each $i = 1,\ldots,n$

$F_i = F_{i-1}(\alpha)$ where $\frac{\alpha'}{\alpha} \in F_{i-1}$

(i.e. F_i is generated by an exponential of an integral over F_{i-1})

or

$F_i = F_{i-1}(\alpha)$ where $\alpha' \in F_{i-1}$

(i.e. F_i is generated by an integral over F_{i-1})

or

F_i is finite algebraic over F_{i-1}.

(i.e. $F_i = F_{i-1}(\alpha)$ and α satisfies a polynomial of the form $a_0+a_1\alpha+\cdots+a_n\alpha^n=0$ where the a_i are in F_{i-1} and are not all zero)

The algorithm need only find one Liouvillian solution of the equation because a second solution may be found by the method of reduction of order as follows. The second solution is assumed of the form $z_2 = v \cdot z_1$ where z_1 is the known first solution and v is some function to be determined. Using the differential equation (1.2a), one obtains a first order equation for v which can be solved to give the second solution as

$$z_2 = z_1 \cdot \int \frac{e^{-\int \frac{b}{a}dx}}{z_1^2} dx$$

If the first solution, z_1, is Liouvillian it is clear that the second, z_2, is also Liouvillian and hence all solutions of (1.2a) are Liouvillian (since all solutions are a linear combination of z_1 and z_2).

In order to reduce the original differential equation to a simpler form, the following transformation is made:

$$y(x) = z(x) \cdot e^{-\int \frac{b}{2a} dx}$$

Then (1.2a) becomes

$$y'' - \left(\frac{b^2 + 2ab' - 2ba' - 4ac}{4a^2} \right) y = 0$$

or

$$y'' = ry \tag{1.2b}$$

where

$$r = \frac{b^2 + 2ab' - 2ba' - 4ac}{4a^2}$$

Clearly, if the solutions to (1.2a) are Liouvillian, then so are the solutions to (1.2b).

In what follows, the equation to be solved will be assumed of the form (1.2b). The implementation of the algorithm accepts equations of the form (1.2a), makes the transformation, solves the transformed equation, then transforms the solutions using the inverse transformation

$$z(x) = y(x) \cdot e^{\int \frac{b}{2a} dx}$$

CHAPTER 2

Some Preliminaries

2.1. The Four Cases

The following theorem by Kovacic [3] determines the form that the algorithm will take.

Theorem: For the differential equation $y'' = ry$, $r \in \mathbf{C}(x)$, there are four cases that can occur.

(1) The d.e. has a solution of the form $\eta = e^{\int \omega}$ where $\omega \in \mathbf{C}(x)$.

(2) The d.e. has a solution of the form $\eta = e^{\int \omega}$ where ω is algebraic of degree 2 over $\mathbf{C}(x)$ and case (1) does not hold.

(3) All solutions of the d.e. are algebraic over $\mathbf{C}(x)$ and cases (1) and (2) do not hold. The solutions are of the form $\eta = e^{\int \omega}$ where ω is algebraic of degree 4, 6 or 12 over $\mathbf{C}(x)$.

(4) The d.e. has no Liouvillian solutions.

The remainder of this section will cover the proof of this theorem and the background necessary to understand it.

Let η, ζ be any two independent solutions of the d.e. $y'' = ry$. Define $\overline{G}$ to be the differential extension field of $\mathbf{C}(x)$ generated by η and ζ, i.e. $\overline{G} = \mathbf{C}(x)(\eta,\eta',\zeta,\zeta')$. (Higher derivatives of η and ζ are not necessary since $\eta'' = r\eta \in \overline{G}$, $\eta''' = r'\eta + r\eta' \in \overline{G}$, etc.)

Now, the *Galois group of the differential equation* is the Galois group of $\overline{G}$ over $\mathbf{C}(x)$, and is denoted $G = G(\overline{G}/\mathbf{C}(x))$. G is the group of all differential automorphisms of $\overline{G}$ leaving $\mathbf{C}(x)$ elementwise fixed.

Recall that an automorphism of a group H is an isomorphism from H to itself. A differential automorphism is an automorphism that commutes with $'$ (the differentiation operator).

This means that G is the group of all automorphisms $\sigma : \overline{G} \to \overline{G}$ such that $\sigma(a') = (\sigma a)'$ for all $a \in \overline{G}$, and $\sigma f = f$ for all $f \in \mathbf{C}(x)$.

The Galois group, G, is isomorphic to a subgroup of $GL(2,\mathbf{C})$, the group of all 2×2 invertible matrices with complex coefficients, i.e. each $\sigma \in G$ corresponds to a matrix $\begin{bmatrix} a_\sigma & b_\sigma \\ c_\sigma & d_\sigma \end{bmatrix}$ where $a_\sigma,\ b_\sigma,\ c_\sigma,\ d_\sigma \in \mathbf{C}$. This correspondence occurs as follows. Because η and ζ are solutions of $y'' = ry$, and because any $\sigma \in G$ is a differential automorphism, then

$$(\sigma(\eta))'' = \sigma(\eta'') = \sigma(r\eta) = \sigma r \cdot \sigma\eta = r\sigma\eta$$

and hence $\sigma\eta$ must be a solution of the d.e. too. Further, $\sigma\eta$ must be a linear combination of η and ζ, since every solution of the d.e. is a linear combination of any two independent solutions of the d.e. We may then write

$$\sigma\eta = a_\sigma \cdot \eta + b_\sigma \cdot \zeta \quad a_\sigma, b_\sigma \in \mathbf{C}$$

Following the same arguments

$$\sigma\zeta = c_\sigma \cdot \eta + d_\sigma \cdot \zeta \quad c_\sigma, d_\sigma \in \mathbf{C}$$

Combining these two results we have

$$\begin{bmatrix} a_\sigma & b_\sigma \\ c_\sigma & d_\sigma \end{bmatrix} \begin{bmatrix} \eta \\ \zeta \end{bmatrix} = \begin{bmatrix} a_\sigma\eta + b_\sigma\zeta \\ c_\sigma\eta + d_\sigma\zeta \end{bmatrix}$$

and σ clearly corresponds to the matrix $\begin{bmatrix} a_\sigma & b_\sigma \\ c_\sigma & d_\sigma \end{bmatrix}$.

Using the Wronskian of η and ζ, we can show that G is isomorphic to a subgroup of $SL(2,\mathbf{C})$ $(\subset GL(2,\mathbf{C}))$, the group of 2×2 invertible matrices with determinant 1. The Wronskian, W, of η and ζ is by definition $W = \eta\zeta' - \eta'\zeta$. Take the derivative of W and get

$$W' = \eta'\zeta' + \eta\zeta'' - \eta'\zeta' - \eta''\zeta = \eta\zeta'' - \eta''\zeta = \eta r\zeta - r\eta\zeta = 0$$

Hence W must be a constant and so for any $\sigma \in G$, $\sigma W = W$ (because $W \in \mathbf{C}(x)$ and σ, by definition, leaves $\mathbf{C}(x)$ fixed). This implies

$$\sigma W = \sigma(\eta\zeta' - \eta'\zeta) = \sigma\eta(\sigma\zeta)' - (\sigma\eta)'\sigma\zeta$$

$$= (a_\sigma\eta + b_\sigma\zeta)(c_\sigma\eta' + d_\sigma\zeta') - (a_\sigma\eta' + b_\sigma\zeta')(c_\sigma\eta + d_\sigma\zeta)$$

$$= (a_\sigma d_\sigma - b_\sigma c_\sigma)(\eta\zeta' - \eta'\zeta) = (a_\sigma d_\sigma - b_\sigma c_\sigma)W$$

and thus $a_\sigma d_\sigma - b_\sigma c_\sigma = \det \sigma = 1$.

We will now state two facts without proof. The Galois group of the d.e., G, (relative to η and ζ), is (isomorphic to) an algebraic subgroup of $SL(2,\mathbf{C})$. (This is a fundamental fact from Picard-Vessiot theory. A proof may be found in [3].) Recall that any subgroup K of $GL(2,\mathbf{C})$ is said to be an *algebraic group* if there exist a finite number of polynomials $P_1, \ldots, P_n$, where each $P_i \in \mathbf{C}[X_1,X_2,X_3,X_4]$, such that a matrix $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ is an element of K iff $P_1(a,b,c,d) = \cdots = P_n(a,b,c,d) = 0$.

Further, for any algebraic subgroup of $SL(2,\mathbf{C})$ the following lemma holds. (See [2] for the proof of this lemma.)

Lemma: If G is an algebraic subgroup of $SL(2,\mathbf{C})$ then one of the following four cases can occur.

(1) G is triangulisable, i.e. there exists an $x \in G$ such that for every $g \in G$, xgx^{-1} is a triangular matrix. We can assume that xgx^{-1} is a lower triangular matrix and hence of the form $\begin{bmatrix} a & 0 \\ b & a^{-1} \end{bmatrix}$ with a and $b \in \mathbf{C}$. (Recall that G is a subgroup of $SL(2,\mathbf{C})$ and hence the determinant of xgx^{-1} must be 1.)

(2) G is conjugate to a subgroup of D^+ where

$$D^+ = \left\{ \begin{bmatrix} c & 0 \\ 0 & c^{-1} \end{bmatrix} \mid c \in \mathbf{C}, c \neq 0 \right\} \cup \left\{ \begin{bmatrix} 0 & c \\ -c^{-1} & 0 \end{bmatrix} \mid c \in \mathbf{C}, c \neq 0 \right\}$$

and case (1) does not hold, i.e. there exists an $x \in G$ such that for every $g \in G$, xgx^{-1} is either a diagonal matrix or a skew-diagonal matrix but there is no $x \in G$ such that all xgx^{-1} are triangular (this includes strictly diagonal too). (Note that the determinant of each xgx^{-1} is 1 since the determinant of each matrix in D^+ is 1.)

(3) G is finite and cases (1) and (2) do not hold.

(4) $G = SL(2,\mathbf{C})$, i.e. G is the infinite group of all 2×2 matrices with determinant 1.

What do we have now? We know that G, the Galois group of the d.e., is (isomorphic to) an algebraic subgroup of $SL(2,\mathbf{C})$. We also know that any algebraic subgroup of $SL(2,\mathbf{C})$ satisfies the above lemma. We can now apply the lemma to the

Galois group of the d.e. and see what relevance it has to the solutions of the d.e.

In case (1), G is triangulisable. Assume $x \in G$ has been found and every matrix conjugated to a lower triangular matrix. (This is equivalent to changing the basis of the vector space or picking two different independent solutions $\overline{\eta}$ and $\overline{\zeta}$.) Then every $\sigma \in G$ is of the form $\begin{bmatrix} a_\sigma & 0 \\ c_\sigma & a_\sigma^{-1} \end{bmatrix}$ $a_\sigma, c_\sigma \in \mathbf{C}$ and maps η to $\sigma\eta = a_\sigma\eta$.

Now if we set $\omega = \dfrac{\eta'}{\eta}$ (or equivalently, $\eta = e^{\int\omega}$), then

$$\sigma\omega = \sigma\left(\frac{\eta'}{\eta}\right) = \frac{(\sigma\eta)'}{\sigma\eta} = \frac{a_\sigma\eta'}{a_\sigma\eta} = \frac{\eta'}{\eta} = \omega$$

and hence $\omega \in \mathbf{C}(x)$. This is case (1) of the original theorem; the d.e. has a solution of the form $\eta = e^{\int\omega}$ where $\omega \in \mathbf{C}(x)$.

In case (2), G is conjugate to a subgroup of D^+. Assume we have conjugated G and every $\sigma \in G$ is of the form $\begin{bmatrix} a_\sigma & 0 \\ 0 & a_\sigma^{-1} \end{bmatrix}$ or $\begin{bmatrix} 0 & b_\sigma \\ -b_\sigma^{-1} & 0 \end{bmatrix}$ so that either $\sigma\eta = a_\sigma\eta$, $\sigma\zeta = a_\sigma^{-1}\zeta$, or $\sigma\eta = b_\sigma\zeta$, $\sigma\zeta = -b_\sigma^{-1}\eta$. (Note that in either case, $\sigma(\eta^2\zeta^2) = \eta^2\zeta^2$, so that $\eta^2\zeta^2 \in \mathbf{C}(x)$.) If we set $\omega = \dfrac{\eta'}{\eta}$ $(\eta = e^{\int\omega})$ and $\phi = \dfrac{\zeta'}{\zeta}$, then either $\sigma\omega = \omega$ and $\sigma\phi = \phi$, or $\sigma\omega = \phi$ and $\sigma\phi = \omega$. Minimally both cases are handled by $\sigma^2\omega = \omega$ or $\sigma^2\omega - \omega = 0$ so ω satisfies a polynomial of degree 2 over $\mathbf{C}(x)$, hence is algebraic of degree 2 over $\mathbf{C}(x)$. This is case (2) of the original theorem; the d.e. has a solution of the form $\eta = e^{\int\omega}$ where ω is algebraic of degree 2 over $\mathbf{C}(x)$.

In case (3), G is a finite group, i.e. there are only a finite number of automorphisms, $\sigma_1, \sigma_2, \ldots, \sigma_n$. Look at any elementary symmetric function of the functions $\sigma_1\eta$, $\sigma_2\eta$, $\cdots$ $\sigma_n\eta$, e.g. $\sum\sigma_i\eta = \sigma_1\eta + \sigma_2\eta + \cdots + \sigma_n\eta$. For any $\sigma_j \in G$, $\sigma_j\left(\sum\sigma_i\eta\right) = \sum\sigma_i\eta$ because $\sigma_j \cdot \sigma_i \in G$ for all σ_i (because G is a group and hence is closed). Hence, $\sum\sigma_i\eta = f(x) \in \mathbf{C}(x)$ and η satisfies $\sigma_1\eta + \cdots + \sigma_n\eta - f(x) = 0$ and is algebraic over $\mathbf{C}(x)$. A similar argument holds for ζ so that since η and ζ are algebraic over $\mathbf{C}(x)$, all solutions of the d.e. are algebraic over $\mathbf{C}(x)$.

To be more specific about the nature of G in case (3), we will state the following theorem without proof. (See Kovacic [3] for details.)

Theorem: If K is a finite subgroup of $SL(2,\mathbf{C})$ then either

(1) K is conjugate to a subgroup of D^+.

(2) The order of K is 24.

(3) The order of K is 48.

(4) The order of K is 120.

Clearly, the first case of this theorem is a subcase of case (2). This means that for case (3), G has order 24, 48 or 120 only, and hence the order of η over $\mathbf{C}(x)$ is 24, 48 or 120 respectively.

In each of these cases, the following functions of η and ζ are known to be in $\mathbf{C}(x)$; if G has order 24 then $(\eta^4 + 8\eta\zeta^3)^2 \in \mathbf{C}(x)$, if G has order 48 then $(\eta^5\zeta - \eta\zeta^5)^2 \in \mathbf{C}(x)$, and if G has order 120 then $\eta^{11}\zeta - 11\eta^6\zeta^6 - \eta\zeta^{11} \in C(x)$. (See [3] for proofs of these statements.)

In case (4), $G = SL(2,\mathbf{C})$. We want to show that in this case, the d.e. has no Liouvillian solution. We will assume the contrary and force a contradiction.

Assume the d.e. has one Liouvillian solution. Then a second solution, obtained by the method of reduction of order, must also be Liouvillian and hence all solutions of the d.e. must be Liouvillian (because all solutions are a linear combination of the two independent solutions). Clearly, $\overline{G} = \mathbf{C}(x)(\eta,\eta',\zeta,\zeta')$ must be contained in a Liouvillian field. This implies that the component of the identity of G, G°, must be solvable.

The component of the identity of any group is the largest connected subgroup of the group containing the identity. Recall that a point set is connected if any two points in the set can be joined by a segmental arc all of whose points belong to the point set.

A group H is said to be solvable (in the Galois theory sense) if $H = H_0 \supset H_1 \supset \cdots \supset H_m = \{e\}$ where each H_{i+1} is normal in H_i, each factor group H_i/H_{i+1} is abelian and e is the identity element of H.

If $G = SL(2,\mathbf{C})$, then $G^\circ = SL(2,\mathbf{C})$ and hence $SL(2,\mathbf{C})$ must be solvable. But $SL(2,\mathbf{C})$ is not solvable and a contradiction has been shown. Hence, the original assumption must be false and the d.e. has no Liouvillian solutions. This is case (4) of the original theorem.

2.2. Form of the Algorithm

At this point, we have that for the d.e. $y'' = ry$, the solution is of the form $e^{\int \omega}$ with either $\omega \in \mathbf{C}$, ω algebraic of degree 2 over $\mathbf{C}(x)$, or ω algebraic of degree 4, 6 or

12 over $\mathbf{C}(x)$, or there is no closed-form solution. The algorithm to solve the given d.e. now takes the following form.

apply sub-algorithm to find a solution of form $\eta = e^{\int \omega}$ where $\omega \in \mathbf{C}(x)$
if success then
    RETURN(solution);
fi;
apply sub-algorithm to find a solution of form $\eta = e^{\int \omega}$ where ω is algebraic of degree 2 over $\mathbf{C}(x)$
if success then
    RETURN(solution);
fi;
apply sub-algorithm to find a solution of form $\eta = e^{\int \omega}$ where ω is algebraic of degree 4, 6 or 12 over $\mathbf{C}(x)$
if success then
    RETURN(solution);
fi;
FAIL();

Each of the three sub-algorithms must be such that if a solution of the required form exists, then it will always be found. Only then can one guarantee that if all three sub-algorithms fail, then there is no Liouvillian solution.

2.3. Necessary Conditions

In order to reduce the work involved in solving the d.e., Kovacic has determined some conditions on the function r that must be true in order for each of the first three cases to be possible, i.e. for a Liouvillian solution to exist. If we can determine using these conditions that a case is not possible, then the sub-algorithm for that case need not be attempted.

These conditions are necessary but not sufficient; i.e. if the conditions for a given case do not hold then the corresponding sub-algorithm need not be tried as it will certainly fail, but if the conditions do hold then the sub-algorithm may or may not succeed.

In order to understand the necessary conditions and their development, some facts from complex analysis are needed.

Recall that any analytic function, f, of a complex variable z, can be expanded about any point a in the complex plane in a Laurent series as follows.

$$f(z) = a_0 + a_1(z-a) + a_2(z-a)^2 + \cdots + \frac{a_{-1}}{z-a} + \frac{a_{-2}}{(z-a)^2} + \cdots$$

The analytic part of the given expansion is $a_0 + a_1(z-a) + \cdots$; the principal part is $\frac{a_{-1}}{z-a} + \frac{a_{-2}}{(z-a)^2} + \cdots$. By definition, a is a *pole of $f(z)$ of order n* if the last term of the principal part of the Laurent series expansion of f about a is $\frac{a_{-n}}{(z-a)^n}$. Equivalently, if f is a rational function, a is a pole of $f(z)$ of order n if it is a root of the denominator of f of multiplicity n.

Further, the *order of f at ∞* is defined to be the order of ∞ as a zero of $f(z)$, i.e. the order of 0 as a pole of $f(\frac{1}{z})$. Equivalently, if f is a rational function, then the order of f at ∞ is the degree of the denominator minus the degree of the numerator.

The following theorem regarding the necessary conditions for the three cases may now be stated.

Theorem: For the d.e. $y'' = ry$ the following conditions are necessary for the respective cases to hold, i.e. for a Liouvillian solution of the specified form to exist.

(1) Every pole of r has order 1 or even order. The order of r at ∞ is even or greater than 2.

(2) r has at least one pole of either order 2 or odd order greater than 2.

(3) No pole of r has order greater than 2. The order of r at ∞ is at least 2. If the partial fraction expansion of r is $r = \sum_i \frac{\alpha_i}{(x-c_i)^2} + \sum_j \frac{\beta_j}{x-d_j}$ then $\sqrt{1+4\alpha_i} \in \mathbf{Q}$ for each i, $\sum_j \beta_j = 0$, and $\sqrt{1+4\gamma} \in \mathbf{Q}$ where $\gamma = \sum_i \alpha_i + \sum_j \beta_j d_j$.

The following is a summary of the proof of this theorem. It uses several ideas that are explained in more detail in the proof of the algorithm itself (see section 3.2.).

In case (1), the d.e. has a solution of the form $\eta = e^{\int \omega}$, where $\omega \in \mathbf{C}(x)$. This implies that

$$\omega' + \omega^2 = r \tag{2.3a}$$

(See section 3.2. for a proof of this statement.) Since both r and $\omega \in \mathbf{C}(x)$, they can

be expanded in Laurent series about any point c in the complex plane as follows.

$$\omega = b(x-c)^{\mu} + \text{higher powers of } x-c, \quad \mu \in \mathbf{Z},\ b \neq 0 \tag{2.3b}$$

$$r = \alpha(x-c)^{\nu} + \text{higher powers of } x-c, \quad \nu \in \mathbf{Z},\ \alpha \neq 0 \tag{2.3c}$$

Substituting (2.3b) and (2.3c) into (2.3a) we get

$$\mu b(x-c)^{\mu-1} + \cdots + b^2(x-c)^{2\mu} + \cdots = \alpha(x-c)^{\nu} + \cdots$$

We want to demonstrate that if c is a pole of r (i.e. $\nu < 0$), then its order is either 1 or even. If we assume ν is not -1 or -2, we can show it must be even. Assume $\nu \leq -3$, then matching coefficients of the lowest power of $x-c$ above gives $\nu \geq \min(\mu-1, 2\mu)$. With $\nu \leq -3$, this implies that $\mu < -1$ and $2\mu < \mu-1$. Because $b^2 \neq 0$ (by assumption), then $\nu = 2\mu$, i.e. ν is even as required.

This also demonstrates that if r has a pole of order $\nu = 2\mu \geq 4$ at c, then ω has a pole of order μ at c. This fact will be used in the proof of the algorithm in section 3.2.1.

The proof of the conditions on the order of r at ∞ is exactly analogous and is done by expanding r and ω at ∞.

In case (2), the d.e. has a solution of the form $\eta = e^{\int \omega}$, ω algebraic over $\mathbf{C}(x)$ of degree 2. The Galois group of the d.e., G, is conjugate to a subgroup of D^+ so that for every $\sigma \in G$ either $\sigma\eta = a_\sigma\eta$, $\sigma\zeta = a_\sigma^{-1}\zeta$ or $\sigma\eta = b_\sigma\zeta$, $\sigma\zeta = -b_\sigma^{-1}\eta$. In either case, $\sigma(\eta^2\zeta^2) = \eta^2\zeta^2$ so $\eta^2\zeta^2 \in \mathbf{C}(x)$. Also $\eta\zeta \notin \mathbf{C}(x)$ because if it were, we would have $\sigma(\eta\zeta) = \eta\zeta = a_\sigma\eta \cdot a_\sigma^{-1}\zeta$ and G would be a diagonal matrix with a_σ and a_σ^{-1} on the diagonal (i.e. the case $\sigma\eta = b_\sigma\zeta$, $\sigma\zeta = -b_\sigma^{-1}\eta$ could not occur).

Hence we can write $\eta^2\zeta^2$ as $\prod(x-c_i)^{e_i}$, $e_i \in \mathbf{Z}$, where at least one of the e_i must be odd. Assume $\eta^2\zeta^2 = (x-c)^e\prod(x-c_i)^{e_i}$ with e odd. Let $\phi = \dfrac{(\eta\zeta)'}{\eta\zeta} = \dfrac{\frac{1}{2}(\eta^2\zeta^2)'}{(\eta^2\zeta^2)}$. Because $\eta'' = r\eta$ and $\zeta'' = r\zeta$, then

$$\phi'' + 3\phi\phi' + \phi^3 = 4r\phi + 2r' \tag{2.3d}$$

Expand both r and ϕ in Laurent series about c. Then

$$\phi = \frac{\frac{1}{2}e}{x-c} + \text{polynomial in } x-c \tag{2.3e}$$

$$r = \alpha(x-c)^{\nu} + \text{higher powers of } x-c \tag{2.3f}$$

and substitute (2.3e) and (2.3f) into (2.3d) to get

$$\frac{e}{(x-c)^3} + \cdots + \frac{-\frac{3}{4}e^2}{(x-c)^3} + \cdots + \frac{\frac{1}{8}e^3}{(x-c)^3} + \cdots = 2\alpha(e+\nu)(x-c)^{\nu-1} + \cdots$$

If $\nu > -2$, then $e - \frac{3}{4}e^2 + \frac{1}{8}e^3 = 0$ and $e = 0, 2, 4$. But e must be odd, so $\nu \leq -2$. If $\nu < -2$, then $2\alpha(e+\nu) = 0$ and $e = -\nu$ so that ν is odd. Hence either $\nu = -2$ or $\nu < -2$ and odd, i.e. r has a pole of either order 2 or odd order > 2.

In case (3), η is algebraic over $\mathbf{C}(x)$ so it can be expanded in a Puiseaux series (Laurent series with fractional exponents) about any point c in the complex plane. Also, η is a solution of the d.e. so

$$\eta'' = r\eta \tag{2.3g}$$

Expand η and r about c

$$\eta = a(x-c)^{\mu} + \text{higher powers of } x-c, \quad a \in \mathbf{C}, \quad a \neq 0, \quad \mu \in \mathbf{Q} \tag{2.3h}$$

$$r = \alpha(x-c)^{\nu} + \text{higher powers of } x-c, \quad \alpha \neq 0, \quad \nu \in \mathbf{Z} \tag{2.3i}$$

and substitute (2.3h) and (2.3i) into (2.3g) to get

$$a\mu(\mu-1)(x-c)^{\mu-2} + \cdots = \alpha a(x-c)^{\mu+\nu} + \cdots$$

The lowest order term on the right, being composed of the product of the lowest order terms of η and r, cannot be zero, so $\mu + \nu \geq \mu - 2$ and $\nu \geq -2$, i.e. the poles of r have order 1 or 2.

If $\nu = -2$, then matching coefficients of $(x-c)^{\mu-2}$ on both sides gives $\alpha = \mu(\mu-1)$ or $\mu = \frac{1}{2} \pm \frac{1}{2}\sqrt{1+4\alpha}$. Because $\mu \in \mathbf{Q}$, by assumption, then $\sqrt{1+4\alpha} \in \mathbf{Q}$ and the partial fraction expansion of r must be

$$r = \sum_i \frac{\alpha_i}{(x-c_i)^2} + \sum_j \frac{\beta_j}{x-d_j} + \text{polynomial}$$

with $\sqrt{1+4\alpha_i} \in \mathbf{Q}$ for each i.

The remainder of the conditions are obtained in an exactly analogous manner by expanding r and η about ∞ and substituting into $\eta'' = r\eta$.

CHAPTER 3

Kovacic's Algorithm and Proof

3.1. Kovacic's Algorithm

In his original paper, Kovacic describes and proves the sub-algorithms for each of the three cases separately. We will exploit the similarities in the three algorithms and describe them together. This will make it easier to follow the unification Saunders does for his variant of the algorithm. Proofs of the three sub-algorithms will be done separately in section 3.2.

The goal of the algorithm will be to determine the minimal polynomial for ω. Since $\eta = e^{\int \omega}$ in all cases, this determines a solution of the d.e. (In some cases, we may not be able to obtain an explicit expression for ω.)

We will first determine a function $\phi = \phi(\eta,\zeta)$. Then, in each case, the minimal polynomial for ω is written in terms of ϕ (and r). The form of ϕ as a function of η and ζ will be determined by the invariant of the Galois group of the d.e. in each case, where the *invariant of the Galois group of the differential equation* is defined to be that function of η and ζ that is kept invariant by all σ in the group and hence is in $\mathbf{C}(x)$. Recall that in case (1), $\frac{\eta'}{\eta}$ is invariant, in case (2) $\eta^2\zeta^2$ is invariant, and in case (3) $(\eta^4 + 8\eta\zeta^3)^2$, $(\eta^5\zeta - \eta\zeta^5)^2$ and $\eta^{11}\zeta - 11\eta^6\zeta^6 - \eta\zeta^{11}$ are invariant.

In all cases, ϕ will be written as $\phi = \theta + \frac{P'}{P}$. The main component of θ is a sum $\sum_{c\in\Gamma} \frac{e_c}{x-c}$, where Γ is the set of poles c of r. A major part of the work of the algorithm will be determining possible values for the e_c's.

The function P will be a polynomial whose roots are *ordinary points of* r (i.e. not poles). We will not determine the roots of P (i.e. the poles of $\frac{P'}{P}$) explicitly; rather we will determine a possible degree d of P in terms of the e_c's and e_∞ (determined from the expansion of r at ∞). Then, we can determine the coefficients of P using an equation relating P, θ and r.

We will need to consider all combinations of possibilities for the e_c's and e_∞ to get a solution for P. If a solution can be found for P, we will have found the proper

combination of e_c's and e_∞ and will have determined θ exactly (since it is a function of the e_c's) and hence ϕ and ω.

The algorithm is divided into 3 steps. As a preliminary stage, the set Γ of the poles of r is computed. Also, the degree of r at infinity is computed; it will be required in the computation of e_∞.

The quantity n will be the degree of ω over $\mathbf{C}(x)$ in each case; for case (1) $n = 1$, for case (2) $n = 2$ and for case (3) $n = 4$, 6 or 12.

Step (1)

For each c in Γ define a set E_c of possible values of e_c as follows:

(a) If c is a pole of r of order 1 then

case (1) – $E_c = \{1\}$

case (2) – $E_c = \{4\}$

case (3) – $E_c = \{12\}$

(b) If c is a pole of r of order 2 and b is the coefficient of $\frac{1}{(x-c)^2}$ in the partial fraction expansion of r then

case (1) – $E_c = \{\frac{1}{2} + k\sqrt{1+4b} \mid k = \pm\frac{1}{2}\}$

case (2) – $E_c = \{2 + 2k\sqrt{1+4b} \mid k = 0, \pm 1\} \cap \mathbf{Z}$

case (3) – $E_c = \{6 + \frac{12}{n}k\sqrt{1+4b} \mid k = 0, \pm 1, \ldots, \pm\frac{n}{2}\} \cap \mathbf{Z}$

(c) If c is a pole of r of order $v > 2$ then

case (1) – $E_c = \{\frac{1}{4}v + k\frac{b}{a} \mid k = \pm\frac{1}{2}\}$

where $[\sqrt{r}]_c$ is the sum of terms involving $\frac{1}{(x-c)^i}$ for $i = 2, \ldots, \frac{v}{2}$ in the Laurent series expansion of $\sqrt{r}$ at c, a is the coefficient of $\frac{1}{(x-c)^{\frac{v}{2}}}$ in $[\sqrt{r}]_c$, and b is the coefficient of $\frac{1}{(x-c)^{\frac{v}{2}+1}}$ in $r - ([\sqrt{r}]_c)^2$

case (2) – $E_c = \{v\}$

case (3) – $E_c = \{\}$ (since there are no poles of order > 2 in case (3))

Also define a set E_∞ as follows.

(a) If the order of r at $\infty > 2$ then

case (1) – $E_\infty = \{\frac{1}{2} + k \mid k = \pm\frac{1}{2}\}$

case (2) – $E_\infty = \{2 + 2k \mid k = 0, \pm 1\}$

case (3) – $E_\infty = \{6 + \frac{12}{n}k \mid k = 0, \pm 1, \ldots, \pm\frac{n}{2}\}$

(b) If the order of r at $\infty = 2$ and b is the coefficient of $\frac{1}{x^2}$ in the Laurent series expansion of r at ∞ then

case(1) – $E_\infty = \{\frac{1}{2} + k\sqrt{1+4b} \mid k = \pm\frac{1}{2}\}$

case(2) – $E_\infty = \{2 + 2k\sqrt{1+4b} \mid k = 0, \pm 1\} \cap \mathbf{Z}$

case(3) – $E_\infty = \{6 + \frac{12}{n}k\sqrt{1+4b} \mid k = 0, \pm 1, \ldots, \pm\frac{n}{2}\} \cap \mathbf{Z}$

(c) If the order of r at $\infty = v < 2$ then

case (1) – $E_\infty = \{\frac{1}{4}v + k\frac{b}{a} \mid k = \pm\frac{1}{2}\}$

where $[\sqrt{r}]_\infty$ is the sum of terms involving x^i for $i = \frac{-v}{2}, \ldots, 0$ in the Laurent series expansion of $\sqrt{r}$ at ∞, a is the coefficient of $x^{\frac{-v}{2}}$ in $[\sqrt{r}]_\infty$, and b is the coefficient of $x^{\frac{-v}{2}-1}$ in $r - ([\sqrt{r}]_\infty)^2$

case (2) – $E_\infty = \{v\}$

case (3) – $E_\infty = \{\}$ (since the order of r at ∞ is ≥ 2 in case(3))

Step (2)

Consider all possible tuples $(e_{c_1}, e_{c_2}, \ldots, e_{c_n}, e_\infty)$, where the c_i are the distinct elements of Γ and each e_{c_i} and e_∞ is an element of the corresponding set E_{c_i} and E_∞ respectively. (For case (2), we may discard a tuple if all of its coordinates are even.)

Form the quantity d as follows:

case (1) – $d = e_\infty - \sum_{c\in\Gamma} e_c$

case (2) – $d = \frac{1}{2}\left(e_\infty - \sum_{c\in\Gamma} e_c\right)$

case (3) – $d = \frac{n}{12}\left(e_\infty - \sum_{c\in\Gamma} e_c\right)$

If d is a non-negative integer, retain the tuple for step (3); otherwise, discard the tuple.

Step (3)

For each tuple retained from step (2), form the rational function θ as follows:

case (1) – $\theta = \sum_{c\in\Gamma}\left(\frac{e_c}{x-c} + s(c)[\sqrt{r}]_c\right) + s(\infty)[\sqrt{r}]_\infty$

where $[\sqrt{r}]_c$ is computed only for poles c of order > 2 and $s(c)$ is the

sign of k in the corresponding e_c in the tuple, and $[\sqrt{r}]_\infty$ is computed only if the order of r at ∞ is < 2 and $s(\infty)$ is the sign of k in the corresponding e_∞ in the tuple.

case (2) – $\theta = \frac{1}{2}\sum_{c\in\Gamma}\frac{e_c}{x-c}$

case (3) – $\theta = \frac{n}{12}\sum_{c\in\Gamma}\frac{e_c}{x-c}$

Now search for a polynomial P of degree d defined by the following equations for $n+2$ polynomials P_i, $i = n, n-1, \ldots, 0, -1$. (These are somewhat different from those given in Kovacic's paper; the rationale for them is given in section 3.2.3.)

cases (1), (2) and (3) – $P_n = -P$

$$P_{i-1} = -P_i' - \theta P_i - (n-i)(i+1)rP_{i+1} \qquad i = n, n-1, \ldots, 0$$

and

$P_{-1} = 0$ (identically)

In each case, P is computed by constructing the polynomial of degree d with undetermined coefficients, substituting into the above equations and solving the final equation $P_{-1} = 0$ for the undetermined coefficients. If the polynomial P exists, then compute ω as follows. (Again, this is slightly different from Kovacic's paper; see section 3.2.3.)

cases (1), (2) and (3) – ω is a solution of $\sum_{i=0}^{n}\frac{P_i}{(n-i)!}\omega^i = 0$

(Note that it may be impossible to obtain an explicit solution for ω.)

Then, $\eta = e^{\int\omega}$ is a solution of the d.e. If no polynomial P exists for any tuple retained from step (2), then the case cannot hold.

3.2. Proof of Kovacic's Algorithm

In the proofs that follow, we will use the following fact several times. The d.e. $y'' = ry$ has a solution of the form $\eta = e^{\int\omega}$ iff ω satisfies the Riccati equation $\omega' + \omega^2 = r$.

The first half of the proof is as follows. Because η is a solution to $y'' = ry$, $\eta'' = r\eta$. Since $\eta = e^{\int\omega}$, $\eta' = \omega e^{\int\omega}$ and $\eta'' = \omega' e^{\int\omega} + \omega^2 e^{\int\omega}$. Thus, $\omega'\eta + \omega^2\eta = r\eta$ and dividing through by $\eta = e^{\int\omega}$ (since it is not zero), we have $\omega' + \omega^2 = r$.

In the converse case, define $\eta = e^{\int\omega}$, i.e. $\omega = \frac{\eta'}{\eta}$. Then, $\omega' = \frac{\eta''}{\eta} - \frac{(\eta')^2}{\eta^2}$ and $\omega^2 = \frac{(\eta')^2}{\eta^2}$. Since $\omega' + \omega^2 = r$, by assumption,

$$\frac{\eta''}{\eta} - \frac{(\eta')^2}{\eta^2} + \frac{(\eta')^2}{\eta^2} = \frac{\eta''}{\eta} = r,$$

proving $\eta'' = r\eta$ and that η is a solution of $y'' = ry$.

3.2.1. Proof of Algorithm for Case (1)

In case (1), we are searching for a solution to the d.e. $y'' = ry$ of the form $\eta = e^{\int \omega}$, $\omega \in \mathbf{C}(x)$. Recall from section 3.1 that we are looking for a function $\phi = \theta + \frac{P'}{P}$. In this case, $\omega = \phi$.

Since $\omega \in \mathbf{C}(x)$, it can be expanded in a Laurent series about any point in the complex plane. The algorithm proceeds by determining the partial fraction expansion of ω and is proved using the Laurent series expansion of r and the Riccati equation

$$\omega' + \omega^2 = r \tag{3.2.1a}$$

We will write the Laurent series expansion of ω about a pole c of r as

$$\omega = \sum_{i=2}^{\mu} \frac{a_i}{(x-c)^i} + \frac{e_c}{x-c} + \sum_{j=0}^{\infty} b_i(x-c)^i$$

Further, we will not need to determine the a_i's and b_i's explicitly, so we will set $[\omega]_c = \sum_{i=2}^{\mu} \frac{a_i}{(x-c)^i}$ and $\overline{\omega}_c = \sum_{j=0}^{\infty} b_i(x-c)^i$. Then

$$\omega = [\omega]_c + \frac{e_c}{x-c} + \overline{\omega}_c = \sum_{i=2}^{\mu} \frac{a_i}{(x-c)^i} + \frac{e_c}{x-c} + \sum_{j=0}^{\infty} b_i(x-c)^i \tag{3.2.1b}$$

We will call $[\omega]_c + \frac{e_c}{x-c}$, the "component at c" of the expansion of ω.

The major task of the algorithm is to determine parts of ω, i.e. the e_c and $[\omega]_c$ and the polynomial remainder part $\overline{\omega}_c$.

Now we know that the poles of r are of either order 1, order 2 or even order ≥ 4, from the necessary conditions for case (1).

Suppose c is a pole of r of order 1. Then

$$r = \frac{\#}{x-c} + \text{polynomial in } x-c \tag{3.2.1c}$$

(We will use # as a placeholder, to denote a complex constant whose value is unknown and unimportant.) Substitute (3.2.1b) and (3.2.1c) into the Riccati equation (3.2.1a) and get

$$\frac{-\mu a_\mu}{(x-c)^{\mu+1}} + \cdots + \frac{a_\mu^2}{(x-c)^{2\mu}} + \cdots = \frac{\#}{x-c} + \cdots$$

Since, by assumption, $a_\mu \neq 0$, matching coefficients of $\frac{1}{x-c}$ on both sides gives $\min(\mu+1,2\mu) = 1 \rightarrow \mu \leq 0$ and hence $[\omega]_c = \sum_{i=2}^{\mu} \frac{a_i}{(x-c)^i} = 0$ (because μ is supposed to be ≥ 2) and $\omega = \frac{e_c}{x-c} + \bar{\omega}_c$.

Use this expression and the Riccati equation (3.2.1a) again and get

$$\frac{-e_c}{(x-c)^2} + \bar{\omega}_c' + \frac{e_c^2}{(x-c)^2} + \frac{2e_c\bar{\omega}_c}{x-c} + \bar{\omega}_c^2 = \frac{\#}{x-c} + \cdots$$

Matching coefficients of $\frac{1}{(x-c)^2}$ on both sides gives $-e_c + e_c^2 = 0$, i.e. e_c is either 0 or 1. The solution $e_c = 0$ can be eliminated since in that case, the right hand side of the above equation has a pole at c and the left hand side does not.

Hence, if c is a pole of r of order 1, then the component at c of ω is

$$\frac{\mathbf{e_c}}{\mathbf{x-c}} \qquad \mathbf{e_c = 1}$$

Now suppose that c is a pole of r of order 2. Then

$$r = \frac{b}{(x-c)^2} + \frac{\#}{x-c} + \cdots \tag{3.2.1d}$$

Substitute (3.2.1b) and (3.2.1d) into the Riccati equation (3.2.1a) (as before) and get

$$\frac{-\mu a_\mu}{(x-c)^{\mu+1}} + \cdots + \frac{a_\mu^2}{(x-c)^{2\mu}} + \cdots = \frac{b}{(x-c)^2} + \frac{\#}{x-c} + \cdots$$

As before, match coefficients of $\frac{1}{(x-c)^2}$ on both sides and get $\min(\mu+1,2\mu) = 2 \rightarrow \mu \leq 1$ and again $[\omega]_c = 0$ (because μ should be ≥ 2) and $\omega = \frac{e_c}{x-c} + \bar{\omega}_c$.

Now use this expression and the Riccati equation (3.2.1a) again and get

$$\frac{-e_c}{(x-c)^2} + (\bar{\omega}_c)' + \frac{e_c^2}{(x-c)^2} + \frac{2e_c\bar{\omega}_c}{x-c} + \bar{\omega}_c^2 = \frac{b}{(x-c)^2} + \frac{\#}{x-c} + \cdots$$

Matching coefficients of $\frac{1}{(x-c)^2}$ on both sides gives $-e_c + e_c^2 = b$, i.e. two possibilities for e_c, $e_c = \frac{1}{2} + \frac{1}{2}\sqrt{1+4b}$ or $e_c = \frac{1}{2} - \frac{1}{2}\sqrt{1+4b}$.

Hence, if c is a pole of r of order 2, then the component at c of ω is

$$\frac{e_c}{x-c} \qquad e_c = \frac{1}{2} \pm \frac{1}{2}\sqrt{1+4b}$$

Now suppose that c is a pole of r of order $\nu = 2\tau \geq 4$. From the proof of the necessary conditions for case (1) (see section 2.3.) we have that ω must have a pole of order $\frac{\nu}{2}$ at c, i.e. $[\omega]_c = \sum_{i=2}^{\frac{\nu}{2}} \frac{a_i}{(x-c)^i}$.

As defined in the statement of the algorithm (section 3.1)

$$[\sqrt{r}]_c = \frac{a}{(x-c)^{\frac{\nu}{2}}} + \cdots + \frac{\#}{(x-c)^2} \qquad (3.2.1e)$$

If we now define $\bar{r}_c = \sqrt{r} - [\sqrt{r}]_c$ then $r = (\bar{r}_c + [\sqrt{r}]_c)^2 = \bar{r}_c^{\,2} + 2\bar{r}_c[\sqrt{r}]_c + ([\sqrt{r}]_c)^2$ and

$$r - ([\sqrt{r}]_c)^2 = \bar{r}_c^{\,2} + 2\bar{r}_c[\sqrt{r}]_c \qquad (3.2.1f)$$

Using (3.2.1e) and (3.2.1f) and the Riccati equation (3.2.1a), we can show (after several lines of not very interesting or important algebra) that

$$([\omega]_c + [\sqrt{r}]_c) \cdot ([\omega]_c - [\sqrt{r}]_c)$$

$$= -[\omega]_c' + \frac{e_c}{(x-c)^2} - \bar{\omega}_c' + r - ([\sqrt{r}]_c)^2 - \frac{2e_c[\omega]_c}{x-c} - \frac{e_c^{\,2}}{(x-c)^2} - \frac{2e_c\bar{\omega}_c}{x-c} - 2\bar{\omega}_c[\omega]_c - \bar{\omega}_c^{\,2}$$

The left hand side of this equation has only terms involving $\frac{1}{(x-c)^i}$ for $i = 4, \ldots, \nu$. The right hand side has terms involving $\frac{1}{(x-c)^i}$ for $i = 1, \ldots, \frac{\nu}{2}+1$ and polynomials in $x-c$. Because there are no terms with $\frac{1}{(x-c)^i}$ for $i = \frac{\nu}{2}+2, \ldots, \nu$ on the right hand side, the left hand side must be equal to zero and hence either $[\omega]_c = [\sqrt{r}]_c$ or $[\omega]_c = -[\sqrt{r}]_c$, and $\omega = \pm[\sqrt{r}]_c + \frac{e_c}{x-c} + \bar{\omega}_c$.

Use this expression and the Riccati equation (3.2.1a) again and (after several more lines of not very interesting algebra) get

$$\frac{\pm a\cdot\frac{\nu}{2}}{(x-c)^{\frac{\nu}{2}+1}} + \cdots + \frac{e_c}{(x-c)^2} - \bar{\omega}_c' + \frac{b}{(x-c)^{\frac{\nu}{2}+1}} + \cdots$$

$$\frac{\mp 2ae_c}{(x-c)^{\frac{\nu}{2}+1}} - \frac{{e_c}^2}{(x-c)^2} - \frac{2e_c\bar{\omega}_c}{x-c} \mp \frac{2\bar{\omega}_c a}{(x-c)^{\frac{\nu}{2}}} + \cdots = 0$$

Matching coefficients of $\frac{1}{(x-c)^{\frac{\nu}{2}+1}}$ on both sides gives $\pm a\cdot\frac{\nu}{2} + b \mp 2ae_c = 0$ and $e_c = \frac{1}{2}(\frac{\nu}{2} + \frac{b}{a})$ or $e_c = \frac{1}{2}(\frac{\nu}{2} - \frac{b}{a})$.

Hence, if c is a pole of r of even order $\nu \geq 4$, then the component at c of the partial fraction expansion of ω is

$$\frac{\mathbf{e_c}}{\mathbf{x-c}} + [\sqrt{\mathbf{r}}]_{\mathbf{c}} \qquad \mathbf{e_c} = \tfrac{1}{2}(\tfrac{\nu}{2} + \tfrac{\mathbf{b}}{\mathbf{a}})$$

or

$$\frac{\mathbf{e_c}}{\mathbf{x-c}} - [\sqrt{\mathbf{r}}]_{\mathbf{c}} \qquad \mathbf{e_c} = \tfrac{1}{2}(\tfrac{\nu}{2} - \tfrac{\mathbf{b}}{\mathbf{a}})$$

Now, look at g, an ordinary point of r, i.e. not a pole, so that r is a polynomial in $x-g$. Expanding ω about g and using the Riccati equation (3.2.1a) and arguments similar to the first case, we can show that $\omega = \frac{f}{x-g}$ + polynomial in $x-g$ where f is either 0 or 1.

Collecting what we have so far, if Γ is the set of poles of r, then

$$\omega = [\omega]_c + \frac{e_c}{x-c} + \bar{\omega}_c = \sum_{c\in\Gamma}\left(\frac{e_c}{x-c} \pm [\sqrt{r}]_c\right) + \sum_{i=1}^{d}\frac{1}{x-g_i} + R$$

where $[\sqrt{r}]_c = 0$ if c is not a pole of order ≥ 4 and R is a polynomial in $\mathbf{C}[x]$.

We now determine the polynomial part R using the expansion of ω about ∞, namely

$$\omega = R + \frac{e_\infty}{x} + \text{lower powers of } x \qquad (3.2.1g)$$

Using arguments analogous to the previous cases we obtain $e_\infty = 0, 1$, $R = 0$ if $o(\infty) > 2$, $e_\infty = \frac{1}{2} \pm \frac{1}{2}\sqrt{1+4b}$, $R = 0$ if $o(\infty) = 2$ and $e_\infty = \frac{1}{2}(\frac{\nu}{2} \pm \frac{b}{a})$, $R = \pm[\sqrt{r}]_\infty$ if $o(\infty) = \nu \leq 0$.

Hence,

$$\omega = \sum_{c\in\Gamma}\left(\frac{e_c}{x-c} + s(c)[\sqrt{r}]_c\right) + s(\infty)[\sqrt{r}]_\infty + \sum_{i=1}^{d}\frac{1}{x-g_i} \qquad \textbf{(3.2.1h)}$$

where $s(c)$ is + or - according to the sign in the corresponding e_c, $s(\infty)$ is + or - according to the sign in e_∞, $[\sqrt{r}]_c = 0$ if c is not a pole of r of order ≥ 4 and $[\sqrt{r}]_\infty = 0$ if $o(\infty) \geq 2$. By expanding (3.2.1h) about ∞ and setting it equal to (3.2.1g), we obtain the equation $e_\infty = \sum_{c\in\Gamma} e_c + \sum_{i-1}^{d} 1$ and hence an expression for d in terms of the e_c's and e_∞, namely $d = e_\infty - \sum_{c\in\Gamma} e_c$.

If we now set $P = \prod_{i=1}^{d}(x-g_i)$ (note that d is the degree of P) so that $\frac{P'}{P} = \sum_{i=1}^{d}\frac{1}{x-g_i}$ and if $\theta = \sum_{c\in\Gamma}\left(\frac{e_c}{x-c} \pm [\sqrt{r}]_c\right) \pm [\sqrt{r}]_\infty$, we have

$$\omega = \phi = \theta + \frac{P'}{P} \qquad (3.2.1i)$$

All of θ is known; we require a method of determining P.

Using the Riccati equation (3.2.1a) again, and $\omega = \theta + \frac{P'}{P}$, we obtain

$$\omega' = \theta' + \frac{P\cdot P'' - P'^2}{P^2}, \qquad \omega^2 = \theta^2 + \frac{2\theta P'}{P} + \frac{P'^2}{P^2}$$

$$P'' + 2\theta P' + P(\theta' + \theta^2 - r) = 0 \qquad (3.2.1j)$$

We have that if ω satisfies the Riccati equation $\omega' + \omega^2 = r$ then P satisfies (3.2.1j). We can verify that if P satisfies (3.2.1j), then ω satisfies the Riccati equation and hence $\eta = e^{\int\omega}$ satisfies the d.e.

$$\omega' + \omega^2 = \theta' + \frac{P'' - P'^2}{P^2} + \theta^2 + \frac{2\theta P'}{P} + \frac{P'^2}{P^2} = \frac{P'' + 2\theta P' + P(\theta' + \theta^2)}{P} = \frac{Pr}{P} = r$$

This completes the proof of the correctness of the algorithm for case (1).

3.2.2. Proof of Algorithm for Case (2)

In case (2), we are searching for a solution to the d.e. $y'' = ry$ of the form $\eta = e^{\int\omega}$ where ω is algebraic of degree 2 over $\mathbf{C}(x)$. The Galois group of the d.e. is conjugate to a subgroup of

$$D^+ = \left\{\begin{pmatrix} c & 0 \\ 0 & c^{-1}\end{pmatrix} \mid c \in \mathbf{C}(x),\ c \neq 0\right\} \cup \left\{\begin{pmatrix} 0 & c \\ -c^{-1} & 0\end{pmatrix} \mid c \in \mathbf{C},\ c \neq 0\right\}$$

and $\eta^2\zeta^2$ is an invariant of the group. Hence, $\eta^2\zeta^2 \in \mathbf{C}(x)$ and $\eta\zeta \notin \mathbf{C}(x)$ (or else we would have case (1)). Therefore we can write

$$\eta^2\zeta^2 = \text{constant} \cdot \prod_{c\in\Gamma}(x-c)^{e_c} \prod_{i=1}^{m}(x-g_i)^{f_i}$$

and

$$\phi = \frac{(\eta\zeta)'}{\eta\zeta} = \frac{1}{2}\frac{(\eta^2\zeta^2)'}{\eta^2\zeta^2} = \frac{1}{2}\sum_{c\in\Gamma}\frac{e_c}{x-c} + \frac{1}{2}\sum_{i=1}^{m}\frac{f_i}{x-g_i} \tag{3.2.2a}$$

The task of the algorithm will be to determine the e_c and f_i. (We do not need to determine the g_i explicitly.) Once ϕ is determined, there is a quadratic equation depending on ϕ that determines ω and hence the solution.

Because η and ζ are solutions to the d.e., i.e. $\eta'' = r\eta$ and $\zeta'' = r\zeta$,

$$\phi'' + 3\phi\phi' + \phi^3 = 4r\phi + 2r' \tag{3.2.2b}$$

This has given us a relationship between ϕ (and the e_c and f_i), and r, a known function.

We can now determine the e_c by looking at the poles of r and the Laurent series expansion of r and ϕ about these poles.

Suppose c is a pole of r of order 1. Then

$$r = \frac{\alpha}{x-c} + \text{polynomial in } x-c \tag{3.2.2c}$$

and

$$\phi = \frac{\frac{1}{2}e_c}{x-c} + k + \text{polynomial in } x-c, \quad k \in \mathbf{C} \tag{3.2.2d}$$

Substituting (3.2.2c) and (3.2.2d) into (3.2.2b), we obtain

$$\frac{e_c}{(x-c)^3} + \cdots + \frac{-\frac{3}{4}e_c^2}{(x-c)^3} + \cdots + \frac{-\frac{3}{2}e_c k}{(x-c)^2} + \cdots + \frac{\frac{1}{8}e_c^3}{(x-c)^3} + \frac{\frac{3}{4}e_c^2 k}{(x-c)^2} + \cdots$$
$$= \frac{2\alpha e_c}{(x-c)^2} + \cdots + \frac{-2\alpha}{(x-c)^2} + \cdots$$

Matching coefficients of $\frac{1}{(x-c)^3}$ on both sides gives $e_c - \frac{3}{4}e_c^2 + \frac{1}{8}e_c^3 = 0$ $\rightarrow e_c = 0, 2, 4$. Matching coefficients of $\frac{1}{(x-c)^2}$ on both sides gives $-\frac{3}{2}e_c k + \frac{3}{4}e_c^2 k = 2\alpha e_c - 2\alpha$. Because $\alpha \neq 0$, $e_c \neq 0, 2$.

Hence, if c is a pole of r of order 1, then

$$\mathbf{e_c = 4}$$

Now suppose c is a pole of r of order 2. Then

$$r = \frac{b}{(x-c)^2} + \frac{\#}{x-c} + \text{polynomial in } x-c \qquad (3.2.2e)$$

$$\phi = \frac{\frac{1}{2}e_c}{x-c} + \text{polynomial in } x-c \qquad (3.2.2f)$$

Substituting (3.2.2e) and (3.2.2f) into (3.2.2b) we get

$$\frac{e_c}{(x-c)^3} + \cdots + \frac{-\frac{3}{4}e_c^2}{(x-c)^3} + \cdots + \frac{\frac{1}{8}e_c^3}{(x-c)^3} = \frac{2be_c}{(x-c)^3} + \cdots + \frac{-4b}{(x-c)^3} + \cdots$$

Matching coefficients of $\frac{1}{(x-c)^3}$ on both sides gives $e_c - \frac{3}{4}e_c^2 + \frac{1}{8}e_c^3 = 2be_c - 4b$ or three possibilities for e_c, $e_c = 2,\ 2 \pm 2\sqrt{1+4b}$. Since e_c is assumed an integer, non-integral solutions for e_c may be discarded.

Hence, if c is a pole of r of order 2 then

$$\mathbf{e_c = 2,\ 2 \pm 2\sqrt{1+4b} \in Z}$$

Now suppose that c is a pole of r of order $\nu > 2$. Then

$$r = \frac{\alpha}{(x-c)^\nu} + \text{higher powers of } x-c \qquad (3.2.2g)$$

$$\phi = \frac{\frac{1}{2}e_c}{x-c} + \text{polynomial in } x-c \qquad (3.2.2h)$$

Substitute (3.2.2g) and (3.2.2h) into (3.2.2b) and get

$$\frac{e_c}{(x-c)^3} + \cdots + \frac{-\frac{3}{4}e_c^2}{(x-c)^3} + \cdots + \frac{\frac{1}{8}e_c^3}{(x-c)^3} + \cdots = \frac{2\alpha e_c}{(x-c)^{\nu+1}} + \cdots + \frac{-2\alpha\nu}{(x-c)^{\nu+1}}$$

Since $\nu > 2$, $\nu+1 > 3$ and $2\alpha e_c - 2\alpha\nu = 0 \rightarrow e_c = \nu$.

Hence if c is a pole of r of order $\nu > 2$ then

$$\mathbf{e_c = \nu}$$

Now look at the g_i which are poles of ϕ but ordinary points of r. Then

$$r = \text{polynomial in } x-g_i \qquad (3.2.2i)$$

and

$$\phi = \frac{\frac{1}{2}f_i}{x-g_i} + k + \text{polynomial in } x-g_i, \quad k \in \mathbf{C} \tag{3.2.2j}$$

Substitute (3.2.2i) and (3.2.2j) into (3.2.2b) and get

$$\frac{f_i}{(x-g_i)^3} + \cdots + \frac{-\frac{3}{4}f_i^2}{(x-g_i)^3} + \frac{-\frac{3}{2}f_i g}{(x-g_i)^2} + \cdots + \frac{\frac{1}{8}f_i^3}{(x-g_i)^3} + \frac{\frac{3}{4}f_i^2 g}{(x-g_i)^2} + \cdots$$

$$= \frac{\#}{x-g_i} + \cdots$$

Since there are no terms in $\frac{1}{(x-g_i)^3}$ on the right hand side, $f_i - \frac{3}{4}f_i^2 + \frac{1}{8}f_i^3 = 0$ $\rightarrow f_i = 0, 2, 4$; hence all the f_i in ϕ are even.

Collecting what we have so far, $\eta^2\zeta^2 = \text{constant} \cdot \prod_{c \in \Gamma}(x-c)^{e_c} \cdot P^2$ where $P \in \mathbf{C}[x]$ and $P^2 = \prod_{i=1}^{m}(x-g_i)^{f_i}$.

We can now use the expansion of ϕ about ∞, namely

$$\phi = \frac{\frac{1}{2}e_\infty}{x} + \text{lower powers of } x \tag{3.2.2k}$$

and arguments exactly analogous to the previous cases, to obtain $e_\infty = 0, 2, 4$ if $o(\infty) > 2$, $e_\infty = 2, 2 \pm 2\sqrt{1+4b}$ if $o(\infty) = 2$, and $e_\infty = \nu$ if $o(\infty) = \nu < 2$.

By expanding (3.2.2a) about ∞, setting it equal to (3.2.2k) and extracting the coefficient of $\frac{1}{x}$ on both sides, we can obtain the following equation, $\frac{1}{2}e_\infty = \frac{1}{2}\sum_{c \in \Gamma} e_c + \frac{1}{2}\sum_{i=1}^{m} f_i$. If d is the degree of P, then $2d = \sum_{i=1}^{m} f_i$ so that $d = \frac{1}{2}\left(e_\infty - \sum_{c \in \Gamma} e_c\right)$ (an expression in terms of the e_c's and e_∞).

If we now let $\theta = \frac{1}{2}\sum_{c \in \Gamma} \frac{e_c}{x-c}$, then $\phi = \theta + \frac{P'}{P}$. Use this expression and (3.2.2b) and obtain

$$P''' + 3\theta P' + (3\theta^2 + 3\theta' - 4r')P' + (\theta'' + 3\theta\theta' + \theta^3 - 4r\theta - 2r')P = 0 \tag{3.2.2l}$$

We still don't have ω, the objective of the algorithm. Kovacic introduces the following equation, algebraic in ω,

$$\omega^2 - \phi\omega + \frac{1}{2}\phi' + \frac{1}{2}\phi^2 - r = 0 \tag{3.2.2m}$$

We can verify that if ω is a solution of this equation and case (2) holds, then ω

satisfies $\omega' + \omega^2 = r$ and hence $\eta = e^{\int \omega}$ satisfies the d.e. $y'' = ry$.

If we differentiate (3.2.2m) we get

$$(2\omega - \phi)\omega' = \phi'\omega - \tfrac{1}{2}\phi'' - \phi\phi' + r'$$

From (3.2.2m), we have that $\omega^2 - r = \phi\omega - \frac{1}{2}\phi' - \frac{1}{2}\phi^2$ so that

$$(2\omega - \phi)(\omega' + \omega^2 - r) = -\tfrac{1}{2}(\phi'' + 3\phi\phi' + \phi^3 - 4r\phi - 2r') = -\tfrac{1}{2} \cdot (3.2.2b) = 0$$

so that either $2\omega - \phi = 0$ or $\omega' + \omega^2 - r = 0$. Now $2\omega - \phi$ cannot be zero, since in that case $\omega = \frac{1}{2}\phi \in \mathbf{C}(x)$ and that is covered in case (1), assumed to fail for case (2).

Hence, $\omega' + \omega^2 = r$ and $\eta = e^{\int \omega}$ is a solution of the d.e. This proves the correctness of the algorithm for case (2).

3.2.3. Proof of Algorithm for Case (3)

In case (3), we are searching for an algebraic solution, η, of the d.e. $y'' = ry$ and as in previous cases, we determine it by computing the minimal polynomial for $\omega = \dfrac{\eta'}{\eta}$ ($\eta = e^{\int \omega}$). As stated in section 2.1., the order of the Galois group of the d.e. in this case is 24, 48 or 120; the following theorem says that the degree of the corresponding ω over $\mathbf{C}(x)$ is then 4, 6 or 12 respectively. (See Kovacic [3] for details of the proof.)

Theorem: If $\omega = \dfrac{\eta'}{\eta}$ where η and ζ are solutions of the d.e. and G is the Galois group of the d.e. relative to η and ζ, then if G has order 24, 48 or 120, ω has degree 4, 6 or 12 respectively over $\mathbf{C}(x)$. Also, for any other $\omega_1 = \dfrac{\eta_1'}{\eta_1}$ where η_1 is also a solution of the d.e., the degree of ω_1 over $\mathbf{C}(x)$ is greater than or equal to 4, 6 or 12 respectively, i.e. the ω obtained are minimal.

The algorithm for this case can be carried out in one of two ways: either find a 12th degree polynomial for ω, factor it into irreducible factors and use any of the factors for ω; or try for a 4th degree polynomial for ω, then a 6th degree polynomial for ω, and finally a 12th degree polynomial for ω.

In the implementation, the second alternative was chosen since factoring a 12th degree polynomial is difficult in Maple (if not impossible if it has algebraic extensions). This is also what is done in Saunders' algorithm for the same reasons.

It will be noted in the relevant places in the proof of this case of the algorithm where the algorithms for cases (1) and (2) can be derived from this case. It turns out that the three cases are more similar than Kovacic's paper originally leads us to believe.

The backbone of the algorithm (and the algorithms for cases (1) and (2)) is an n-th order ordinary differential equation for a function ϕ defined in terms of ϕ and r by $a_{-1} = 0$, where a_{-1} is defined by the following equations, denoted $(\#)_n$.

$$\left.\begin{aligned} a_n &= -1 \\ a_{i-1} &= -a_i' - \phi a_i - (n-i)(i+1)r a_{i+1} \qquad i = n, \ldots, 0 \end{aligned}\right\} \qquad (\#)_n$$

(Note that in the formula for a_{n-1} there is no a_{n+1} term since $n-i = 0$ when $i = n$.)

In all three cases, we will construct ϕ as $\phi = \theta + \frac{P'}{P}$, where n is the degree of ω over $\mathbf{C}(x)$ The function θ is constructed as a function of the poles of r and P is then defined in terms of θ and r by the equation $P_{-1} = 0$ where $P_i = P \cdot a_i$.

In case (3), $\theta = \frac{n}{12}\sum_{c \in \Gamma} \frac{e_c}{x-c}$; compare this with $\theta = \sum_{c \in \Gamma}\left(\frac{e_c}{x-c} + s(c)[\sqrt{r}]_c\right) + s(\infty)[\sqrt{r}]_\infty$ in case (1) and $\theta = \frac{1}{2}\sum_{c \in \Gamma} \frac{e_c}{x-c}$ in case (2).

The following three theorems by Kovacic are used in the proof of the algorithm for case (3) but they apply equally well to the algorithms for cases (1) and (2).

Theorem(1): If ϕ is a solution of $a_{-1} = 0$ where a_i is defined by the equations denoted $(\#)_n$, and ω is a solution of the equation

$$\omega^n = \sum_{i=0}^{n-1} \frac{a_i}{(n-i)!}\omega^i$$

then $\eta = e^{\int \omega}$ is a solution of the d.e. $y'' = ry$.

Proof:

We first define the polynomial $A(u)$ in terms of the a_i as

$$A(u) = -u^n + \sum_{i=0}^{n-1} \frac{a_i}{(n-i)!}u^i$$

or

$$A(u) = \sum_{i=0}^{n} \frac{a_i}{(n-i)!} u^i \qquad a_n = -1$$

Kovacic claims that

$$(u^2 - r)\frac{\partial^{k+1}A(u)}{\partial u^{k+1}} = \frac{\partial^{k+1}A(u)}{\partial u^i \partial x} + [(n-2k)u + \phi]\frac{\partial^k A(u)}{\partial u^k} + k(n-k+1)\frac{\partial^{k-1}A(u)}{\partial u^{k-1}} \quad (3.2.3a)$$

for all integer $k \geq 0$. (See [3] for a proof by induction on n of this claim.) From the definition of $A(u)$ and the assumed definition of ω, $A(\omega) = 0$. We will show that $A(\omega) = 0$ implies that $\omega' + \omega^2 = r$ (equivalent to $\eta = e^{\int \omega}$ is a solution of $y'' = ry$), by assuming the contrary and forcing a contradiction.

Because $A(\omega)$ is a constant, $\frac{dA(\omega)}{dx} = 0$. Then

$$\frac{dA(\omega)}{dx} = \frac{\partial A(\omega)}{\partial \omega}\frac{\partial \omega}{\partial x} + \frac{\partial A(\omega)}{\partial x} = 0$$

$$\omega' \frac{\partial A(\omega)}{\partial \omega} + \frac{\partial A(\omega)}{\partial x} = 0$$

$$(\omega' + \omega^2 - r)\frac{\partial A(\omega)}{\partial \omega} = -\frac{\partial A(\omega)}{\partial x} + (\omega^2 - r)\frac{\partial A(\omega)}{\partial \omega}$$

From (3.2.3a) with $k = 0$,

$$(\omega' + \omega^2 - r)\frac{\partial A(\omega)}{\partial \omega} = -\frac{\partial A(\omega)}{\partial x} + (n\omega + \phi)A(\omega) + \frac{\partial A(\omega)}{\partial x} = 0$$

(since $A(\omega) = 0$). Because $\omega' + \omega^2 - r \neq 0$ by assumption, $\frac{\partial A(\omega)}{\partial \omega}$ must be zero. Hence, we have that $\frac{\partial^l A(\omega)}{\partial \omega^l} = 0$ for $l = 0$ and $l = 1$. Now we can use induction to prove $\frac{\partial^{k+1}A(\omega)}{\partial \omega^{k+1}} = 0$. Assume it is true for arbitrary $l = k-1$ and $l = k$, i.e. $\frac{\partial^{k-1}A(\omega)}{\partial \omega^{k-1}} = \frac{\partial^k A(\omega)}{\partial \omega^k} = 0$. If $\frac{\partial^k A(\omega)}{\partial \omega^k} = 0$ then

$$\frac{d}{dx}\left(\frac{\partial^k A(\omega)}{\partial \omega^k}\right) = 0$$

$$\frac{\partial^{k+1}A(\omega)}{\partial \omega^{k+1}}\frac{\partial \omega}{\partial x} + \frac{\partial^{k+1}A(\omega)}{\partial \omega^k \partial x} = 0$$

$$(\omega' + \omega^2 - r)\frac{\partial^{k+1}A(\omega)}{\partial \omega^{k+1}} = -\frac{\partial^{k+1}A(\omega)}{\partial \omega^k \partial x} + (\omega^2 - r)\frac{\partial^{k+1}A(\omega)}{\partial \omega^{k+1}}$$

From (3.2.3a) we have

$$(\omega' + \omega^2 - r)\frac{\partial^{k+1}A(\omega)}{\partial\omega^{k+1}}$$

$$= -\frac{\partial^{k+1}A(\omega)}{\partial\omega^k\partial x} + \frac{\partial^{k+1}A(\omega)}{\partial\omega^k\partial x} + [(n-2k)\omega + \phi]\frac{\partial^k A(\omega)}{\partial\omega^k} + k(n-k+1)\frac{\partial^{k-1}A(\omega)}{\partial\omega^{k-1}}$$

Since $\frac{\partial^k A(\omega)}{\partial\omega^k} = \frac{\partial^{k-1}A(\omega)}{\partial\omega^{k-1}} = 0$ (by assumption), then $(\omega' + \omega^2 - r)\frac{\partial^{k+1}A(\omega)}{\partial\omega^{k+1}} = 0$, and since $\omega' + \omega^2 \neq r$ (by assumption), then $\frac{\partial^{k+1}A(\omega)}{\partial\omega^{k+1}} = 0$.

The desired contradiction then falls out since

$$\frac{\partial^n A(\omega)}{\partial\omega^n} = \frac{\partial^n}{\partial\omega^n}\left(\sum_{i=0}^{n}\frac{a_i}{(n-i)!}\omega^i\right) = \sum_{i=0}^{n}\left(\frac{a_i}{(n-i)!}\frac{\partial(\omega^i)}{\partial\omega^n}\right) = \frac{\partial^n(a_n\omega^n)}{\partial\omega^n} = -n! \neq 0$$

and hence $A(\omega) = 0$ implies $\omega' + \omega^2 = r$ and $\eta = e^{\int\omega}$ is a solution of $y'' = ry$.

Thus if ω satisfies $\omega^n = \sum_{i=0}^{n-1}\frac{a_i}{(n-i)!}\omega^i$ where the a_i correspond to a solution ϕ of $(\#)_n$, then $e^{\int\omega}$ is a solution of $y'' = ry$. This completes the proof of the theorem. □

This theorem implies that if we can construct ϕ and then calculate the corresponding a_i using $(\#)_n$, we can determine an equation for ω and hence a solution of the d.e.

The following theorem says that the equation for ω obtained using the a_i corresponding to ϕ is the minimal polynomial for ω. Further, ϕ is proved to be a function in $\mathbf{C}(x)$.

Theorem(2): If the degree of ω over $\mathbf{C}(x)$ is n, then ϕ is a solution of $a_{-1} = 0$ where a_i is defined by $(\#)_n$, and ϕ is a rational function of x with coefficients in $\mathbf{C}$, i.e. $\phi \in \mathbf{C}(x)$.

Proof:

Let $A(u)$ be a polynomial with coefficients in $\mathbf{C}(x)$, and let $A(u)$ be the minimal polynomial for ω. Let deg $A(u) = n$ so that the degree of ω over $\mathbf{C}(x)$ is n, then $A(u)$ can be of the form

$$A(u) = -u^n + \sum_{i=0}^{n-1}\frac{a_i}{(n-i)!}u^i = \sum_{i=0}^{n}\frac{a_i}{(n-i)!}u^i \qquad a_n = -1$$

Consider the following polynomial $B(u)$

$$B(u) = (r-u^2)\frac{\partial A(u)}{\partial u} + \frac{\partial A(u)}{\partial x} + (nu + \phi)A(u)$$

where $\phi = a_{n-1}$ and $\phi \in \mathbf{C}(x)$. We will show that ϕ satisfies $a_{-1} = 0$ where the a_i are defined by $(\#)_n$, by determining the coefficients of powers of u in $B(u)$.

The u^{n+1} term in $B(u)$ comes from $-u^2\frac{\partial A(u)}{\partial u}$ and $nuA(u)$ and is

$$-u^2 \cdot \frac{na_n u^{n-1}}{0!} + nu \cdot \frac{a_n u^n}{0!} = -na_n u^{n+1} + na_n u^{n+1} = 0$$

The u^n term comes from $-u^2\frac{\partial A(u)}{\partial u}$, $\frac{\partial A(u)}{\partial x}$, $nuA(u)$ and $\phi A(u)$ and is

$$-u^2 \cdot \frac{(n-1)a_{n-1}u^{n-2}}{1!} + \frac{a_n' u^n}{0!} + nu \cdot \frac{a_{n-1}u^{n-1}}{1!} + \phi \cdot \frac{a_n u^n}{0!}$$

$$= -(n-1)a_{n-1}u^n + a_n' u^n + na_{n-1}u^n + \phi a_n u^n$$

$$= (a_{n-1} + a_n' + \phi a_n)u^n = 0$$

since $a_n = -1$, $a_n' = 0$ and since $\phi = a_{n-1}$, $\phi a_n = -a_{n-1}$. Hence, there are no u^{n+1} or u^n terms in $B(u)$ and the degree of $B(u)$ is less than n.

Now $B(\omega) = 0$ as follows

$$B(\omega) = (r-\omega^2)\frac{\partial A(\omega)}{\partial \omega} + \frac{\partial A(\omega)}{\partial x} + (n\omega + \phi)A(\omega)$$

$$= \omega'\frac{\partial A(\omega)}{\partial \omega} + \frac{\partial A(\omega)}{\partial x} + (n\omega + \phi)A(\omega) = \frac{dA(\omega)}{dx} + (n\omega + \phi)A(\omega) = 0$$

since $A(\omega) = 0$ by definition because it is the minimal polynomial for ω. Hence, the coefficients of ω^i in $B(\omega)$ for $i < n$ must all be zero. The ω^i term in $B(\omega)$ is

$$r \cdot \frac{(i+1)a_{i+1}}{(n-i-1)!}\omega^i - \omega^2 \cdot \frac{(i-1)a_{i-1}}{(n-i+1)!}\omega^{i-2} + \frac{a_i'}{(n-i)!}\omega^i + n\omega \cdot \frac{a_{i-1}}{(n-i+1)!}\omega^{i-1} + \phi \cdot \frac{a_i}{(n-i)!}\omega_i$$

$$= \left[\frac{r(i+1)a_{i+1}}{(n-i-1)!} - \frac{(i-1)a_{i-1}}{(n-i+1)!} + \frac{na_{i-1}}{(n-i+1)!} + \frac{\phi a_i}{(n-i)!}\right]\omega^i$$

$$= \frac{1}{(n-i)!}\left[(n-i)(i+1)ra_{i+1} - \frac{(i-1)}{(n-i+1)}a_{i-1} + \frac{na_{i-1}}{(n-i+1)} + a_i' + \phi a_i\right]\omega^i$$

$$= \frac{1}{(n-i)!}\left[(n-i)(i+1)ra_{i+1} + a_{i-1} + a_i' + \phi a_i\right]\omega^i$$

So $(n-i)(i+1)ra_{i+1} + a_{i-1} + a_i' + \phi a_i = 0$ for all $i = 0, \ldots, n$ where $a_{-1} = 0$. These are exactly the equations defining $(\#)_n$ so ϕ is a solution of $a_{-1} = 0$ as

required. This completes the proof of the theorem. □

What do we have now? We have that $\phi \in \mathbf{C}(x)$ is a solution of $a_{-1} = 0$ defined by $(\#)_n$ iff ω satisfying $\omega^n = \sum_{i=0}^{n-1} \frac{a_i}{(n-i)!} w^i$ is a solution of $\omega' + \omega^2 = r$ and ω is algebraic of degree n over $\mathbf{C}(x)$. The algorithms for all three cases use this fact; implicitly in cases (1) and (2), and explicitly in case (3). We progressively try $n = 1$, $n = 2$, $n = 4$, $n = 6$, $n = 12$. At each step we attempt to find a ϕ in $\mathbf{C}(x)$ satisfying $(\#)_n$. If that is possible then the minimal polynomial for ω is $-\omega^n + \sum_{i=0}^{n-1} \frac{a_i}{(n-i)!} \omega^i$. If it is not possible, we proceed to the next value of n. If no value of n produces a ϕ in $\mathbf{C}(x)$ then there is no solution to the d.e. $y'' = ry$.

The following theorem provides a way of building the function ϕ.

Theorem(3): If u is any homogeneous polynomial of degree n in solutions η and ζ of the d.e. then $\phi = \frac{u'}{u}$ is a solution of $(\#)_n$. (Note: for example, $\eta^2\zeta^3 + 3\eta\zeta^4$ is a homogeneous polynomial of degree 5 in solutions of the d.e.)

See [3] for the details of the proof.

We require that ϕ be in $\mathbf{C}(x)$ so that $\phi = \frac{u'}{u} = \frac{\frac{1}{m}(u^m)'}{u^m}$ must be in $\mathbf{C}(x)$. If $\frac{u'}{u}$ is in $\mathbf{C}(x)$ or u^m, $m \geq 1$, is in $\mathbf{C}(x)$ then this requirement is satisfied. The functions u are written in terms of the invariants of the Galois group of the d.e. for each case. (See section 2.2 for the derivation of the invariants.) Recall that the invariant of the Galois group of the d.e. is a function of η and ζ left fixed by all σ in the group, i.e. it is a function in $\mathbf{C}(x)$.

Table 1 - Galois group invariants

n	u	invariant	m
1	η	u'/u	1
2	$\eta\zeta$	u^2	2
4	$\eta^4+8\eta\zeta^3$	u^3	3
6	$\eta^5\zeta-\eta\zeta^5$	u^2	2
12	$\eta^{11}\zeta-11\eta^6\zeta^6-\eta\zeta^{11}$	u	1

We can then write the invariant in terms of the poles of r and certain exponents e_c,

and a polynomial part. Recall that in case (1),

$$\phi = \frac{u'}{u} = \omega = \sum_{c\in\Gamma}\left(\frac{e_c}{x-c} + s(c)[\sqrt{r}]_c\right) + s(\infty)[\sqrt{r}]_\infty + \frac{P'}{P}$$

and in case (2),

$$\phi = \frac{\frac{1}{2}(u^2)'}{u^2} = \frac{\frac{1}{2}(\eta^2\zeta^2)'}{\eta^2\zeta^2} = \frac{1}{2}\sum_{c\in\Gamma}\frac{e_c}{x-c} + \frac{P'}{P}$$

For case (3), we combine the cases $n = 4, 6, 12$ by writing

$$u^{\frac{12}{n}} = \prod_{c\in\Gamma}(x-c)^{e_c}\prod_{i=1}^{m}(x-g_i)^{f_i}$$

$$\phi = \frac{\frac{n}{12}(u^{\frac{12}{n}})'}{u^{\frac{12}{n}}} = \frac{n}{12}\sum_{c\in\Gamma}\frac{e_c}{x-c} + \frac{P'}{P}$$

The e_c's are determined in a manner analogous to the process described for cases (1) and (2), and so will not be covered again here. The derivation is slightly more complicated here because of the parameter n, but basically the same.

The results are as follows. If c is a pole of r of order 1, then $e_c = 12$. If c is a pole of r of order 2, then $e_c = 6 + k\sqrt{1+4b}$, where k is one of $0, \pm 3, \pm 6$ if $n = 4$, k is one of $0, \pm 2, \pm 4, \pm 6$ if $n = 6$, and k is one of $0, \pm 1, \ldots, \pm 6$ if $n = 12$.

Similarly, if the order of r at ∞ is greater than 2, then $e_\infty = 6 + k$ and if the order of r at ∞ is 2, then $e_\infty = 6 + k\sqrt{1+4b}$, where $k = 0, \pm 3, \pm 6$ if $n = 4$, $k = 0, \pm 2, \pm 4, \pm 6$ if $n = 6$, and $k = 0, \pm 1, \ldots, \pm 6$ if $n = 12$.

It can also be shown that $\frac{n}{12}f_i$ is an integer for all i so that for $n = 4$, $u^3 = P^3\prod_{c\in\Gamma}(x-c)^{e_c}$; for $n = 6$, $u^2 = P^2\prod_{c\in\Gamma}(x-c)^{e_c}$; and for $n = 12$, $u = P\prod_{c\in\Gamma}(x-c)^{e_c}$, where the degree of P, $d = \frac{n}{12}\left(e_\infty - \sum_{c\in\Gamma}e_c\right)$, is a non-negative integer.

Compare the above results for case (3) those obtained in cases (1) and (2). The order of the invariant m is defined to be the least integer such that $\phi = \dfrac{\frac{1}{m}(u^m)'}{u^m} \in \mathbf{C}(x)$.

Table 2 - Formulas for e_c

n	m	e_c (c order 1)	e_c (c order 2)	k
1	1	1	$\frac{1}{2}+k\sqrt{1+4b}$	$-\frac{1}{2}, \frac{1}{2}$
2	2	4	$2(1+k\sqrt{1+4b})$	-1, 0, 1
4	3	12	$3(2+k\sqrt{1+4b})$	-2, -1, 0, 1, 2
6	2	12	$2(3+k\sqrt{1+4b})$	-3 , . . . , 3
12	1	12	$6+k\sqrt{1+4b}$	-6 , . . . , 6

The value calculated for e_c if c is a pole of r of order 1 is $n \cdot m$; the value calculated for e_c if c is a pole of r of order 2 is $m(\frac{n}{2} + k\sqrt{1+4b})$ where $k = -\frac{n}{2}, \ldots, \frac{n}{2}$.

So for case (3) we have

$$\phi = \frac{u'}{u} = \frac{\frac{n}{12}(u^{\frac{12}{n}})'}{u^{\frac{12}{n}}} = \frac{n}{12}\sum_{c\in\Gamma}\frac{e_c}{x-c} + \frac{P'}{P}$$

If we set $\theta = \frac{n}{12}\sum_{c\in\Gamma}\frac{e_c}{x-c}$, then $\phi = \theta + \frac{P'}{P}$ as expected.

We now set $P_i = P \cdot a_i$ and demonstrate that the recursive relations for P are correct. Kovacic actually sets $P_i = S^{n-i} \cdot P \cdot a_i$, where $S = \prod_{c\in\Gamma}(x-c)$, with no justification at all. This is completely unnecessary and obscures the similarities of the three cases.

If $P_i = P \cdot a_i$, then $P_n = P \cdot a_n = -P$. Also

$$P_{i-1} = P \cdot a_{i-1} = P \cdot (-a_i' - \phi a_i - (n-i)(i+1)ra_{i+1})$$

$$= -Pa_i' - P\phi a_i - (n-i)(i+1)Pra_{i+1} = -Pa_i' - P\left(\theta + \frac{P'}{P}\right)a_i - (n-i)(i+1)rP_{i+1}$$

$$= -Pa_i' - P\theta a_i - P'a_i - (n-i)(i+1)rP_{i+1} = -P_i' - \theta P_i - (n-i)(i+1)rP_{i+1}$$

Because $a_{-1} = 0$, then $P_{-1} = 0$.

Hence P is defined by the following equations

$$P_n = -P$$

$$P_{i-1} = -P_i' - \theta P_i - (n-i)(i+1)rP_{i+1}$$

and

$$P_{-1} = 0 \quad \text{(identically)}$$

Rewriting the equation for ω in terms of P_i gives

$$0 = -\omega^n + \sum_{i=0}^{n-1} \frac{a_i}{(n-i)!}\omega^i = -P\omega^n + \sum_{i=0}^{n-1} \frac{Pa_i}{(n-i)!}\omega^i = a_n P\omega^n + \sum_{i=0}^{n-1} \frac{P_i}{(n-i)!}\omega^i = \sum_{i=0}^{n} \frac{P_i}{(n-i)!}\omega^i$$

$$\text{i.e.} \quad \sum_{i=0}^{n} \frac{P_i}{(n-i)!}\omega^i = 0$$

We can now verify that these two sets of equations for P and ω are the same as those produced in cases (1) and (2) as follows.

In case (1), $n = 1$.

$$P_1 = -P$$

$$P_0 = -P_1' - \theta P_1 - (1-1)(1+1)rP_2$$

$$= P' + \theta P$$

$$P_{-1} = -P_0' - \theta P_0 - (1-0)(0+1)rP_1$$

$$= -(P' + \theta P)' - \theta(P' + \theta P) - r(-P) = -P'' - \theta' P - \theta P' - \theta P' - \theta^2 P + rP$$

$$= -P'' - 2\theta P' - (\theta' + \theta^2 - r)P$$

because $P_{-1} = a_{-1} \cdot P = 0$ (because $a_{-1} = 0$) then

$$P'' + 2\theta P' + (\theta' + \theta^2 - r)P = 0 \qquad (3.2.3b)$$

Also $\sum_{i=0}^{1} \frac{P_i}{(1-i)!}\omega^i = 0$ so that

$$\frac{P_0}{(1-0)!}\omega^0 + \frac{P_1}{(1-1)!}\omega^1 = 0$$

$$P_0 + P_1\omega = 0$$

$$P' + \theta P - P\omega = 0$$

$$\omega = \frac{P' + \theta P}{P} = \theta + \frac{P'}{P} \qquad (3.2.3c)$$

Note that (3.2.3b) and (3.2.3c) are the same equations as those produced for case (1), namely (3.2.1j) and (3.2.1i) respectively.

In case (2), $n = 2$.

$$P_2 = -P$$

$$P_1 = -P_2' - \theta P_2 - (2-2)(2+1)rP_3$$

$$= P' + \theta P$$

$$P_0 = -P_1' - \theta P_1 - (2-1)(1+1)rP_2$$

$$= -(P' + \theta P)' - \theta(P' + \theta P) - 2r(-P)$$

$$= -P'' - \theta' P - \theta P' - \theta P' - \theta^2 P + 2rP$$

$$= -P'' - 2\theta P' - (\theta' + \theta^2 - 2r)P$$

$$P_{-1} = -P_0' - \theta P_0 - (2-0)(0+1)rP_1$$

$$= -(-P'' - 2\theta P' - (\theta' + \theta^2 - 2r)P)' - \theta(-P'' - 2\theta P' - (\theta' + \theta^2 - 2r)P) - 2r(P' + \theta P)$$

$$= P''' + 2\theta' P' + 2\theta P'' + (\theta' + \theta^2 - 2r)P' + (\theta'' + 2\theta\theta' - 2r')P + \theta P'' + 2\theta^2 P'$$

$$+ \theta\theta' P + \theta^3 P - 2r\theta P - 2rP' - 2r\theta P$$

$$= P''' + 3\theta P'' + (3\theta' + 3\theta^2 - 4r)P' + (\theta'' + 3\theta\theta' + \theta^3 - 4r\theta - 2r')P$$

Because P_{-1} must be zero then

$$P''' + 3\theta P'' + (3\theta' + 3\theta^2 - 4r)P' + (\theta'' + 3\theta\theta' + \theta^3 - 4r\theta - 2r')P = 0 \qquad (3.2.3d)$$

Also $\sum_{i=0}^{2} \frac{P_i}{(2-i)!}\omega^i = 0$, so that

$$\frac{P_0}{(2-0)!}\omega^0 + \frac{P_1}{(2-1)!}\omega^1 + \frac{P_2}{(2-2)!}\omega^2 = 0$$

$$\frac{1}{2}P_0 + P_1\omega + P_2\omega^2 = 0$$

$$\frac{1}{2}(-P'' - 2\theta P' - (\theta' + \theta^2 - 2r)P) + (P' + \theta P)\omega + (-P)\omega^2 = 0$$

$$\omega^2 + \left(\frac{P' + \theta P}{-P}\right)\omega + \frac{1}{2}\left(\frac{-P'' - 2\theta P' - (\theta' + \theta^2 - 2r)P}{-P}\right) = 0$$

$$\omega^2 - \left(\theta + \frac{P'}{P}\right)\omega + \frac{1}{2}\left(\frac{P''}{P} + \frac{2\theta P'}{P} + (\theta' + \theta^2 - 2r)\right) = 0$$

Since $\phi = \theta + \frac{P'}{P}$, $\phi' = \theta' + \frac{P''}{P} - \left(\frac{P'}{P}\right)^2$ then

$$\omega^2 - \phi\omega + \frac{1}{2}(\theta' + \theta^2 - 2r) = 0 \qquad (3.2.3e)$$

Note that (3.2.3d) and (3.2.3e) are the same equations as those produced for case (2), namely (3.2.2l) and (3.2.2m) respectively.

Noticing that the equations for P and ω could be unified for all three cases was of benefit in implementing the algorithm. Saunders' [5] had unified only steps (1) and (2) of the algorithm and implemented step (3) separately for each of the three cases. In the implementation for Maple, step (3) could also be implemented as a single procedure.

CHAPTER 4

Saunders' Algorithm

4.1. Saunders' Modifications to Kovacic's Algorithm

Saunders [5] has produced a modified version of Kovacic's algorithm where most of the algorithm has been unified to avoid implementing each of the cases separately. In his algorithm, only the final portion of step (3) remained to be implemented separately for each case. By noting the similarities of the three cases, it was possible to unify this step as well.

Saunders does not prove that his version of Kovacic's algorithm does in fact correctly implement Kovacic's algorithm and it is certainly not obvious that the two are the same algorithm. In fact, in the course of comparing the two, several bugs were discovered in Saunders' algorithm. A corrected version is presented here and verified correct by comparison with Kovacic's algorithm as given in section 3.1.

Saunders' algorithm is noteworthy in that he has unified the computation of d (the degree of P) and θ in steps (1) and (2) of Kovacic's algorithm. (Recall that we are computing a function $\phi = \theta + \frac{P'}{P}$, then obtaining the minimal polynomial for ω in terms of ϕ and r.) We have now unified the computation of P and ω in step (3) of Kovacic's algorithm; see section 3.2.3 for details.

Saunders' unification is carried out by computing "parts" of d and θ. Recall that in Kovacic's algorithm, d = constant $\cdot \left(e_\infty - \sum_{c \in \Gamma} e_c\right)$ for all three cases, and that the e_c's and e_∞ were of the form $\text{expression}_1 + k \cdot \text{expression}_2$, where $k = -\frac{n}{2}, \ldots, \frac{n}{2}$. Here, we will compute the sum of all the expression_1's in d as e_{fix}, and each expression_2 as e_i, with $i = 0$ for e_∞.

A similar process is carried out for θ. Recall that in Kovacic's algorithm, the main component of θ was the sum $\sum_{c \in \Gamma} \frac{e_c}{x - c}$.

In a preliminary step of the algorithm, we normalise $r = \frac{s}{t}$ where $\gcd(s,t) = 1$ and s and t are polynomials in $\mathbf{C}[x]$. We then perform a square-free decomposition on t and obtain $t = t_1 \cdot t_2^2 \cdot t_3^3 \cdots t_m^m$. We will not have to determine all the poles of r

as in Kovacic's algorithm, only the poles with even order. We also determine the order of r at ∞, $o(\infty) = \deg t - \deg s$.

Next, we determine which of the three cases are possible (equivalently, what degrees of ω over $\mathbf{C}(x)$ are possible) by checking the necessary conditions and creating a list L of possible degrees as follows.

$1 \in L$	if $t_i = 1$ for all odd $i \geq 2$ and $o(\infty)$ is even or > 2
$2 \in L$	if $t_2 \neq 1$ or $t_i \neq 1$ for some odd $i \geq 3$
$4, 6, 12 \in L$	if $t_i = 1$ for all $i > 2$ and $o(\infty) \geq 2$

Then, the algorithm is as follows.

Step (1)
Form parts of d and θ.

(a) $e_{fix} = \frac{1}{4}\min(o(\infty),2) - \frac{1}{4}\deg t - \frac{3}{4}\deg t_1$

$$\theta_{fix} = \frac{\frac{1}{4}t'}{t} + \frac{\frac{3}{4}t_1'}{t_1}$$

(b) find the poles $c_1, \ldots, c_{k_2}$ of r of order 2 (i.e. the roots of t_2)
for i from 1 to k_2 do

b_i = the coefficient of $\frac{1}{(x-c_i)^2}$ in the partial fraction expansion of r

$e_i = \sqrt{1+4b_i}$

$\theta_i = \frac{e_i}{x-c_i}$

od

(c) if $1 \in L$ then find the poles $c_{k_2+1}, \ldots, c_k$ of order $4, 6, 8, \ldots, m$ (i.e. the roots of $t_4, t_6, \ldots, t_m$)
for i from k_2+1 to k do

$[\sqrt{r}]_{c_i}$ = the sum of terms involving $\frac{1}{(x-c_i)^k}$, for $k = 2, \ldots, \frac{\nu}{2}$ and ν the order of the pole c_i, in the Laurent series expansion of $\sqrt{r}$ at c_i

a_i = the coefficient of $(x-c)^{\frac{-\nu}{2}}$ in $[\sqrt{r}]_{c_i}$

b_i = the coefficient of $(x-c_i)^{-\frac{\nu}{2}+1}$ in $r - ([\sqrt{r}]_{c_i})^2$

$e_i = \frac{b_i}{a_i}$

$\theta_i = 2[\sqrt{r}]_{c_i} + \frac{e_i}{x-c_i}$

od

(d) if $o(\infty) > 2$ then

$e_0 = 1$

$\theta_0 = 0$

elsif $o(\infty) = 2$ then

$b_0 =$ the coefficient of $\frac{1}{x^2}$ in the Laurent series expansion of r at ∞

$e_0 = \sqrt{1+4b_0}$

$\theta_0 = 0$

else

if $1 \in L$ then

$[\sqrt{r}]_\infty =$ the sum of terms involving x^i, for $i = \frac{-v}{2}, \ldots, 0$ and $v = o(\infty)$, in the Laurent series expansion of $\sqrt{r}$ at ∞

$a_0 =$ the coefficient of $x^{\frac{-v}{2}}$ in $[\sqrt{r}]_\infty$

$b_0 =$ the coefficient of $x^{\frac{-v}{2}+1}$ in $r - ([\sqrt{r}]_\infty)^2$

$e_0 = \frac{b_0}{a_0}$

$\theta_0 = 2[\sqrt{r}]_\infty$

else

$e_0 = 0$

$\theta_0 = 0$

fi

fi

Step (2)

Form the trial d's and θ's.

for each n in L (in increasing order) do

if $n = 1$ then $m = k$ else $m = k_2$ fi

if $n = 2$ and $o(\infty) < 2$ then

$e_0 = 0$

$\theta_0 = 0$

fi

for all sequences $s = (s_0, \ldots, s_m)$ where $s_i \in \{-\frac{n}{2}, -\frac{n}{2}+1, \ldots, \frac{n}{2}\}$ do

$d = n \cdot e_{fix} + s_0 e_0 - \sum_{i=1}^{m} s_i e_i$

if d is an integer ≥ 0 then

$\theta = n \cdot \theta_{fix} + \sum_{i=0}^{m} s_i \theta_i$
apply step (3) to d and θ
if successful then
RETURN(solution)
fi
fi
od
od
FAIL(); (no solution exists)

Step (3)

Find the polynomial P (if possible) and ω.
form P in terms of undetermined coefficients a_i
$P = a_d x^d + a_{d-1} x^{d-1} + \cdots + a_0$
generate the recursive relations P_i
$P_n = -P$
for i from n by -1 to 0 do
$P_{i-1} = -P_i' - \theta P_i - (n-i)(i+1) r P_{i+1}$
od
solve $P_{-1} = 0$ for the a_i
if a solution exists then
generate the minimal polynomial for ω
minpoly = 0
for i from 0 to n do
minpoly = minpoly + $\frac{P_i}{(n-i)!} \cdot \omega^i$
od
solve minpoly for ω
RETURN(solution as $e^{\int \omega}$)
e.se
not successful
fi

The following corrections were made to Saunders' original algorithm. The expression for d in step (2) was $d = n \cdot e_{fix} - \sum_{i=0}^{m} s_i e_i$. Unless one multiplied the

expressions for e_0 by -1, this expression was incorrect. The corrected expression is more desirable simply because it corresponds more closely to the expression for d in Kovacic's algorithm. (See step (2) in section 3.1.)

Also, it was stated that e_0 and θ_0 were only computed if $o(\infty)$ was ≤ 2, and were never computed if 1 was not in the list L, i.e. if case (1) was not possible. This will be seen to be untrue in the following section.

4.2. Proof of Saunders' (Corrected) Algorithm

In order to verify this algorithm, we will require several identities. If $t = t_1 \cdot t_2^2 \cdot t_3^3 \cdots t_m^m$, with the t_i square-free and pair-wise relatively prime, then

$$\deg t = \deg t_1 + 2 \cdot \deg t_2 + 3 \cdot \deg t_3 + \cdots + m \cdot \deg t_m$$

and

$$\deg t_k = \text{the number of poles of } r \text{ of order } k$$

Also,

$$\frac{t'}{t} = \frac{t_1'}{t_1} + \frac{2t_2'}{t_2} + \cdots + \frac{m \cdot t_m'}{t_m}$$

and

$$\frac{t_k'}{t_k} = \sum_{j=1}^{\deg t_k} \frac{1}{x - a_j}$$

where the a_j , $j = 1, \ldots, \deg t_k$ are the roots of t_k, i.e. the poles of r of order k.

Now, we will verify the corrected version of Saunders' algorithm for each of the three cases of Kovacic's algorithm, i.e. $n = 1$, $n = 2$ and $n = 4$, 6 or 12. In all cases, the verification consists of proving that the d's and θ's computed agree for the two algorithms. The d's and θ's computed by Saunders' algorithm will be denoted d_s and θ_s; those computed by Kovacic's algorithm will be denoted by d_k and θ_k.

Case (1)

$n = 1$

$$d_s = 1 \cdot e_{fix} + s_0 e_0 - \sum_{i=1}^{k} s_i e_i$$

$$= \frac{1}{4}\min(o(\infty),2) - \frac{1}{4}\deg t - \frac{3}{4}\deg t_1 + s_0 e_0 - \sum_{i=1}^{k_2} s_i e_i - \sum_{i=k_2+1}^{k} s_i e_i$$

$$= \frac{1}{4}\min(o(\infty),2) - \deg t_1 - \frac{1}{2}\deg t_2 - \frac{1}{4}\sum_{\substack{\text{poles of}\\ \text{order } \nu>2\\ \nu \text{ even}}} \nu\cdot\deg t_\nu + s_0e_0 - \sum_{i=1}^{k_2} s_ie_i - \sum_{i=k_2+1}^{k} s_ie_i$$

$$= \frac{1}{4}\min(o(\infty),2) + s_0e_0 - \sum_{\substack{c\in\Gamma\\ \text{poles of}\\ \text{order 1}}} 1 - \sum_{\substack{c\in\Gamma\\ \text{poles of}\\ \text{order 2}}} \frac{1}{2} - \sum_{i=1}^{k_2} s_ie_i - \sum_{\substack{c\in\Gamma\\ \text{poles of}\\ \text{order } \nu>2\\ \nu \text{ even}}} \frac{1}{4}\nu - \sum_{i=k_2+1}^{k} s_ie_i$$

$$= \frac{1}{4}\min(o(\infty),2) + s_0e_0 - \sum_{\substack{\text{poles of}\\ \text{order 1}}} 1 - \sum_{\substack{\text{poles of}\\ \text{order 2}}} \left(\frac{1}{2} + s_i\sqrt{1+4b}\right) - \sum_{\substack{\text{poles of}\\ \text{order } \nu>2\\ \nu \text{ even}}} \left(\frac{1}{4}\nu + s_i\frac{b}{a}\right)$$

Compare this formula with the formula for d_k.

$$d_k = e_\infty - \sum_{c\in\Gamma} e_c$$

$$= e_\infty - \sum_{\substack{c\in\Gamma\\ \text{poles of}\\ \text{order 1}}} 1 - \sum_{\substack{c\in\Gamma\\ \text{poles of}\\ \text{order 2}}} \left(\frac{1}{2} + k\sqrt{1+4b}\right) - \sum_{\substack{c\in\Gamma\\ \text{poles of}\\ \text{order } \nu>2\\ \nu \text{ even}}} \left(\frac{1}{4}\nu + k\frac{b}{a}\right) \quad \text{where } k = \pm\frac{1}{2}$$

The three sums are clearly equal in the two formulas since $s_i = -\frac{n}{2}, \ldots, \frac{n}{2} = \pm\frac{1}{2}$ when $n = 1$. It remains to show that $\frac{1}{4}\min(o(\infty),2) + s_0e_0$ corresponds to e_∞. If $o(\infty) > 2$ then

$$\frac{1}{4}\min(o(\infty),2) + s_0e_0 = \frac{1}{4}\cdot 2 + \pm\frac{1}{2}\cdot 1 = \frac{1}{2} \pm \frac{1}{2} = e_\infty$$

and if $o(\infty) = 2$ then

$$\frac{1}{4}\min(o(\infty),2) + s_0e_0 = \frac{1}{4}\cdot 2 + \pm\frac{1}{2}\cdot\sqrt{1+4b} = \frac{1}{2} \pm \frac{1}{2}\sqrt{1+4b} = e_\infty$$

and if $o(\infty) < 2$ then

$$\frac{1}{4}\min(o(\infty),2) + s_0e_0 = \frac{1}{4}\cdot\nu + \pm\frac{1}{2}\cdot\frac{b}{a} = \frac{1}{4}\nu \pm \frac{1}{2}\frac{b}{a} = e_\infty$$

so $d_s = d_k$ for case (1).

Now

$$\theta_s = 1\cdot\theta_{fix} + \sum_{i=0}^{k} s_i\theta_i$$

$$= \frac{1}{4}\frac{t'}{t} + \frac{\frac{3}{4}t_1'}{t_1} + s_0\theta_0 + \sum_{i=1}^{k_2} s_i\theta_i + \sum_{i=k_2+1}^{k} s_i\theta_i$$

$$= \sum_{\substack{c\in\Gamma \\ \text{poles of} \\ \text{order 1}}} \frac{1}{x-c_i} + \sum_{\substack{c\in\Gamma \\ \text{poles of} \\ \text{order 2}}} \frac{\frac{1}{2}}{x-c_i} + \sum_{\substack{c\in\Gamma \\ \text{poles of} \\ \text{order } \nu>2 \\ \nu \text{ even}}} \frac{\frac{1}{4}\nu}{x-c_i} + s_0\theta_0 + \sum_{i=1}^{k_2} s_i\theta_i + \sum_{i=k_2+1}^{k} s_i\theta_i$$

$$= \sum_{\substack{\text{poles of} \\ \text{order 1}}} \frac{1}{x-c_i} + \sum_{\substack{\text{poles of} \\ \text{order 2}}} \frac{\frac{1}{2}+s_ie_i}{x-c_i} + \sum_{\substack{\text{poles of} \\ \text{order } \nu>2 \\ \nu \text{ even}}} \left(\frac{\frac{1}{4}\nu+s_ie_i}{x-c_i} + 2s_i[\sqrt{r}]_{c_i} \right) + s_0\theta_0$$

$$= \sum_{\substack{\text{poles of} \\ \text{order 1}}} \frac{1}{x-c_i} + \sum_{\substack{\text{poles of} \\ \text{order 2}}} \frac{\frac{1}{2}+s_i\sqrt{1+4b}}{x-c_i} + \sum_{\substack{\text{poles of} \\ \text{order } \nu>2 \\ \nu \text{ even}}} \left(\frac{\frac{1}{4}\nu+s_i\frac{b}{a}}{x-c_i} + 2s_i[\sqrt{r}]_{c_i} \right) + s_0\theta_0$$

Since $s_i = -\frac{n}{2}, \ldots, \frac{n}{2} = \pm\frac{1}{2}$

$$\theta_s = \sum_{\substack{\text{poles of} \\ \text{order 1}}} \frac{1}{x-c_i} + \sum_{\substack{\text{poles of} \\ \text{order 2}}} \frac{\frac{1}{2}+s_i\sqrt{1+4b}}{x-c_i} + \sum_{\substack{\text{poles of} \\ \text{order } \nu>2 \\ \nu \text{ even}}} \left(\frac{\frac{1}{4}\nu+s_i\frac{b}{a}}{x-c_i} + \text{sign}(s_i)\cdot[\sqrt{r}]_{c_i} \right) + s_0\theta_0$$

Recall that the formula for θ_k is

$$\theta_k = \sum_{c\in\Gamma} \left(\frac{e_c}{x-c} + s(c)[\sqrt{r}]_c \right) + s(\infty)[\sqrt{r}]_\infty$$

$$= \sum_{\substack{c\in\Gamma \\ \text{poles of} \\ \text{order 1}}} \frac{1}{x-c} + \sum_{\substack{c\in\Gamma \\ \text{poles of} \\ \text{order 2}}} \frac{\frac{1}{2}+k\sqrt{1+4b}}{x-c} + \sum_{\substack{c\in\Gamma \\ \text{poles of} \\ \text{order } \nu>2 \\ \nu \text{ even}}} \left(\frac{\frac{1}{4}\nu+k\frac{b}{a}}{x-c} + s(c)[\sqrt{r}]_c \right) + s(\infty)[\sqrt{r}]_\infty$$

Clearly, the three sums are equal in the two formulas. It remains to show that $s_0\theta_0$ corresponds to $s(\infty)[\sqrt{r}]_\infty$. If $o(\infty) \geq 2$ then

$$s_0\theta_0 = \pm\frac{1}{2}\cdot 0 = 0$$

and if $o(\infty) < 2$ then

$$s_0\theta_0 = \pm\frac{1}{2}\cdot 2[\sqrt{r}]_\infty = s(\infty)[\sqrt{r}]_\infty$$

so $\theta_s = \theta_k$ for case (1) and Saunders' algorithm is verified correct for $n = 1$.

Case (2)

$n = 2$

$$d_s = 2\cdot e_{fix} + s_0e_0 - \sum_{i=1}^{k_2} s_ie_i$$

$$= \tfrac{1}{2}\min(o(\infty),2) - \tfrac{1}{2}\deg t - \tfrac{3}{2}\deg t_1 + s_0e_0 - \sum_{i=1}^{k_2} s_ie_i$$

$$= \tfrac{1}{2}\min(o(\infty),2) - 2\deg t_1 - \deg t_2 - \tfrac{1}{2}\sum_{\nu>2}\nu\deg t_\nu + s_0e_0 - \sum_{i=1}^{k_2} s_ie_i$$

$$= \tfrac{1}{2}\min(o(\infty),2) + s_0e_0 - \sum_{\substack{c\in\Gamma \\ \text{poles of} \\ \text{order 1}}} 2 - \sum_{\substack{c\in\Gamma \\ \text{poles of} \\ \text{order 2}}} 1 - \sum_{i=1}^{k_2} s_ie_i - \sum_{\substack{c\in\Gamma \\ \text{poles of} \\ \text{order } \nu>2}} \tfrac{1}{2}\nu$$

$$= \tfrac{1}{2}\min(o(\infty),2) + s_0e_0 - \sum_{\substack{\text{poles of} \\ \text{order 1}}} 2 - \sum_{\substack{\text{poles of} \\ \text{order 2}}} 1+s_i\sqrt{1+4b} - \sum_{\substack{\text{poles of} \\ \text{order } \nu>2}} \tfrac{1}{2}\nu$$

$$= \tfrac{1}{2}\left(\min(o(\infty),2) + 2s_0e_0 - \sum_{\substack{\text{poles of} \\ \text{order 1}}} 4 - \sum_{\substack{\text{poles of} \\ \text{order 2}}} 2+2s_i\sqrt{1+4b} - \sum_{\substack{\text{poles of} \\ \text{order } \nu>2}} \nu\right)$$

Recall that the formula for d_k is

$$d_k = \tfrac{1}{2}\left(e_\infty - \sum_{c\in\Gamma} e_c\right)$$

$$= \tfrac{1}{2}\left(e_\infty - \sum_{\substack{c\in\Gamma \\ \text{poles of} \\ \text{order 1}}} 4 - \sum_{\substack{c\in\Gamma \\ \text{poles of} \\ \text{order 2}}} 2+2k\sqrt{1+4b} - \sum_{\substack{c\in\Gamma \\ \text{poles of} \\ \text{order } \nu>2}} \nu\right)$$

The three sums are clearly equal in the two formulas since $s_i = -\frac{n}{2}, \ldots, \frac{n}{2} = -1,0,1 = 0,\pm1$ when $n = 2$. It remains to show that $\min(o(\infty),2) + 2s_0e_0$ corresponds to e_∞. If $o(\infty) > 2$ then

$$\min(o(\infty),2) + 2s_0e_0 = 2 + 2\cdot s_0\cdot 1 = 2 + 2s_0 = e_\infty$$

and if $o(\infty) = 2$ then

$$\min(o(\infty),2) + 2s_0e_0 = 2 + 2s_0\sqrt{1+4b} = e_\infty$$

and if $o(\infty) = \nu < 2$ then

$$\min(o(\infty),2) + 2s_0e_0 = \nu + 2\cdot s_0\cdot 0 = \nu = e_\infty$$

so that $d_s = d_k$ for case (2).

Now

$$\theta_s = 2\cdot\theta_{fix} + \sum_{i=0}^{k_2} s_i\theta_i$$

$$= \frac{\frac{1}{2}t'}{t} + \frac{\frac{3}{2}t_1'}{t_1} + s_0\theta_0 + \sum_{i=1}^{k_2} s_i\theta_i$$

$$= \sum_{\substack{c\in\Gamma \\ \text{poles of} \\ \text{order 1}}} \frac{2}{x-c_i} + \sum_{\substack{c\in\Gamma \\ \text{poles of} \\ \text{order 2}}} \frac{1}{x-c_i} + \sum_{\substack{c\in\Gamma \\ \text{poles of} \\ \text{order } \nu>2}} \frac{\frac{1}{2}\nu}{x-c_i} + s_0\theta_0 + \sum_{i=1}^{k_2} \frac{s_i e_i}{x-c_i}$$

$$= \sum_{\substack{\text{poles of} \\ \text{order 1}}} \frac{2}{x-c_i} + \sum_{\substack{\text{poles of} \\ \text{order 2}}} \frac{1+s_i\sqrt{1+4b}}{x-c_i} + \sum_{\substack{\text{poles of} \\ \text{order } \nu>2}} \frac{\frac{1}{2}\nu}{x-c_i} + s_0\theta_0$$

$$= \frac{1}{2}\left(\sum_{\substack{\text{poles of} \\ \text{order 1}}} \frac{4}{x-c_i} + \sum_{\substack{\text{poles of} \\ \text{order 2}}} \frac{2+2s_i\sqrt{1+4b}}{x-c_i} + \sum_{\substack{\text{poles of} \\ \text{order } \nu>2}} \frac{\nu}{x-c_i} + 2s_0\theta_0 \right)$$

Recall that the formula for θ_k is

$$\theta_k = \frac{1}{2}\sum_{c\in\Gamma} \frac{e_c}{x-c}$$

$$= \frac{1}{2}\left(\sum_{\substack{c\in\Gamma \\ \text{poles of} \\ \text{order 1}}} \frac{4}{x-c} + \sum_{\substack{c\in\Gamma \\ \text{poles of} \\ \text{order 2}}} \frac{2+2k\sqrt{1+4b}}{x-c} + \sum_{\substack{c\in\Gamma \\ \text{poles of} \\ \text{order } \nu>2}} \frac{\nu}{x-c} \right)$$

Clearly the three sums are equal in the two formulas. Since $\theta_0 = 0$ in this case regardless of the value of $o(\infty)$, we have $\theta_s = \theta_k$ for case (2) and Saunders' algorithm is verified correct for $n = 2$.

Case (3)
$n = 4,6,12$

$$d_s = n\cdot e_{fix} + s_0e_0 - \sum_{i=1}^{k_2} s_ie_i$$

$$= \frac{n}{4}\min(o(\infty),2) - \frac{n}{4}\deg t - \frac{3n}{4}\deg t_1 + s_0e_0 - \sum_{i=1}^{k_2} s_ie_i$$

$$= \frac{n}{4}\min(o(\infty),2) - n\deg t_1 - \frac{n}{2}\deg t_2 + s_0e_0 - \sum_{i=1}^{k_2} s_ie_i$$

$$= \frac{n}{4}\min(o(\infty),2) + s_0e_0 - \sum_{\substack{c\in\Gamma \\ \text{poles of} \\ \text{order 1}}} n - \sum_{\substack{c\in\Gamma \\ \text{poles of} \\ \text{order 2}}} \frac{n}{2} - \sum_{i=1}^{k_2} s_ie_i$$

$$= \frac{n}{12}\left(3\cdot\min(o(\infty),2) + \frac{12}{n}s_0e_0 - \sum_{\substack{\text{poles of} \\ \text{order 1}}} 12 - \sum_{\substack{\text{poles of} \\ \text{order 2}}} 6+\frac{12}{n}s_i\sqrt{1+4b}\right)$$

Compare this formula with the formula for d_k.

$$d_k = \frac{n}{12}\left(e_\infty - \sum_{c\in\Gamma} e_c\right)$$

$$= \frac{n}{12}\left(e_\infty - \sum_{\substack{c\in\Gamma \\ \text{poles of} \\ \text{order 1}}} 12 - \sum_{\substack{c\in\Gamma \\ \text{poles of} \\ \text{order 2}}} 6+\frac{12}{n}k\sqrt{1+4b}\right)$$

The three sums are clearly equal in the two formulas since $s_i = -\frac{n}{2}, \ldots, \frac{n}{2} = 0, \pm1, \pm2, \ldots, \pm\frac{n}{2}$ when $n = 4$, 6 or 12. It remains to show that $3\cdot\min(o(\infty),2) + \frac{12}{n}s_0e_0$ corresponds to e_∞. If $o(\infty) > 2$ then

$$3\cdot\min(o(\infty),2) + \frac{12}{n}s_0e_0 = 3\cdot2 + \frac{12}{n}\cdot s_0\cdot1 = 6 + \frac{12}{n}s_0 = e_\infty$$

and if $o(\infty) = 2$ then

$$3\cdot\min(o(\infty),2) + \frac{12}{n}s_0e_0 = 3\cdot2 + \frac{12}{n}\cdot s_0\sqrt{1+4b} = 6 + \frac{12}{n}s_0\sqrt{1+4b} = e_\infty$$

and $o(\infty) < 2$ does not occur in case (3).

Now

$$\theta_s = n\cdot\theta_{fix} + \sum_{i=0}^{k_2} s_i\theta_i$$

$$= \frac{\frac{n}{4}t'}{t} + \frac{\frac{3n}{4}t_1'}{t_1} + s_0\theta_0 + \sum_{i=0}^{k_2} s_i\theta_i$$

$$= \sum_{\substack{c\in\Gamma \\ \text{poles of} \\ \text{order 1}}} \frac{n}{x-c_i} + \sum_{\substack{c\in\Gamma \\ \text{poles of} \\ \text{order 2}}} \frac{\frac{n}{2}}{x-c_i} + s_0\theta_0 + \sum_{i=0}^{k_2} s_i\theta_i$$

$$= \sum_{\substack{\text{poles of} \\ \text{order 1}}} \frac{n}{x-c_i} + \sum_{\substack{\text{poles of} \\ \text{order 2}}} \frac{\frac{n}{2}+s_ie_i}{x-c_i} + s_0\theta_0$$

$$= \sum_{\substack{\text{poles of}\\ \text{order 1}}} \frac{n}{x-c_i} + \sum_{\substack{\text{poles of}\\ \text{order 2}}} \frac{\frac{n}{2}+s_i\sqrt{1+4b}}{x-c_i} + s_0\theta_0$$

$$= \frac{n}{12}\left(\sum_{\substack{\text{poles of}\\ \text{order 1}}} \frac{12}{x-c_i} + \sum_{\substack{\text{poles of}\\ \text{order 2}}} \frac{6+\frac{12}{n}s_i\sqrt{1+4b}}{x-c_i} + \frac{12}{n}s_0\theta_0 \right)$$

Recall that the formula for θ_k is

$$\theta_k = \frac{n}{12}\sum_{c\in\Gamma}\frac{e_c}{x-c}$$

$$= \frac{n}{12}\left(\sum_{\substack{c\in\Gamma\\ \text{poles of}\\ \text{order 1}}} \frac{12}{x-c} + \sum_{\substack{c\in\Gamma\\ \text{poles of}\\ \text{order 2}}} \frac{6+\frac{12}{n}k\sqrt{1+4b}}{x-c} \right)$$

Clearly, the sums are equal in the two formulas. Since $\theta_0 = 0$ for any value of $o(\infty)$, then $\theta_s = \theta_k$ for case (3) and Saunders' algorithm is verified correct for $n = 4, 6, 12$.

Hence, the corrected version of Saunders' algorithm is verified correct as presented.

CHAPTER 5

Implementation in Maple

5.1. Details of the Implementation

The implementation in Maple follows Saunders' variant of the algorithm fairly closely. Several subsections of the implementation will be discussed further here.

The square-free decomposition required in the preliminary step of Saunders' algorithm is done using Yun's algorithm (c) as follows. If P is a primitive polynomial, its square-free decomposition $P_1 \cdot P_2^2 \cdot P_3^3 \cdots P_m^m$ is computed by:

$G \leftarrow \gcd\left(P, \frac{dP}{dx}\right)$

$C_1 \leftarrow \frac{P}{G}$

$D_1 \leftarrow \frac{dP}{dx}/G - \frac{dC_1}{dx}$

for i from 1 while $C_1 \neq 1$ do

$P_i \leftarrow \gcd(C_i, D_i)$

$C_{i+1} \leftarrow \frac{C_i}{P_i}$

$D_{i+1} \leftarrow \frac{D_i}{P_i} - \frac{dC_{i+1}}{dx}$

od

A proof that this algorithm does calculate the square-free decomposition of P may be found in [7].

The procedure to find the roots of the parts $t_2, t_4, \ldots, t_m$ was developed to circumvent intrinsic limitations in Maple's *solve* routine. (See section 5.2 for further details.) First, Maple's *solve* is called to determine the roots. If *solve* finds all the roots (the number of roots should be the degree of the input polynomial), then *radsimp* is called to simplify them. Otherwise, *factor* is called in an attempt to reduce the polynomial to factors that can be handled by *solve*. If *factor* cannot produce any factors, the differential equation cannot be solved and the routine fails.

In step 1 of Saunders' algorithm, the calculation of the d_i and of $[\sqrt{r}]_{c_i}$ requires

computation of coefficients of the Laurent series expansion of a rational function at its poles. The function is never explicitly expanded in a Laurent series; instead the method of undetermined coefficients is used. This also requires long division of the rational function to reduce $\frac{s}{t}$ to $squo + \frac{srem}{t}$ where $\deg(srem) < \deg(t)$. The algorithm used for division is algorithm D by Knuth [4].

Then, if a is a root of t with multiplicity m, $t(x) = (x-a)^m \cdot g(x)$ and f is a polynomial in x

$$\frac{srem}{t} = \frac{srem}{(x-a)^m \cdot g} = \frac{A_m}{(x-a)^m} + \frac{A_{m-1}}{(x-a)^{m-1}} + \cdots + \frac{A_1}{x-a} + \frac{f}{g}$$

where

$$A_i = \frac{1}{(m-i)!} \frac{d^{m-i}}{dx^{m-i}} \left(\frac{srem}{g} \right) \Bigg|_{x=a}$$

This algorithm is from the CRC Standard Mathematical Tables [6].

The calculation of $[\sqrt{r}]_{c_i}$ also uses this algorithm for undetermined coefficients. If c_i is a pole of order $v = 2\tau \geq 4$, then $[\sqrt{r}]_{c_i}$ is the sum of terms involving $\frac{1}{(x-c_i)^i}$ for $i = 2,3, \ldots, \tau$ in the Laurent series expansion of $\sqrt{r}$ at c_i. So

$$[\sqrt{r}]_{c_i} = \frac{a_\tau}{(x-c_i)^\tau} + \frac{a_{\tau-1}}{(x-c_i)^{\tau-1}} + \cdots + \frac{a_2}{(x-c_i)^2}$$

and $([\sqrt{r}]_{c_i})^2$ agrees with $(\sqrt{r})^2 = r$ for powers of $\frac{1}{(x-c_i)^k}$, $k = \tau+2, \ldots, 2\tau$. We exploit this fact by squaring $[\sqrt{r}]_{c_i}$ and obtaining expressions equal to the coefficients of $\frac{1}{(x-c_i)^k}$ in r, then using the undetermined coefficients algorithm on r.

$$([\sqrt{r}]_{c_i})^2 = \left(\frac{a_\tau}{(x-c_i)^\tau} + \frac{a_{\tau-1}}{(x-c_i)^{\tau-1}} + \cdots + \frac{a_2}{(x-c_i)^2} \right) = \sum_{i=4}^{v} \left[\frac{1}{(x-c_i)^i} \cdot \left(\sum_{j=i-\tau}^{\tau} a_j \cdot a_{i-j} \right) \right]$$

We first solve for a_τ by noting

$$a_\tau^2 = \text{coefficient of } \frac{1}{(x-c_i)^{2\tau}} \text{ in } r$$

Then each succeeding coefficient, $a_{\tau-1}, a_{\tau-2}, \ldots, a_2$ can be solved for using the equation $\sum_{j=i-\tau}^{\tau} a_j \cdot a_{i-j} =$ coefficient of $\frac{1}{(x-c_i)^i}$ in r and substitution of previously computed a's.

The calculation of $[\sqrt{r}]_\infty$ is very similar. If the order of r at ∞ is $\nu = -2\tau$, then $[\sqrt{r}]_\infty$ is the sum of terms involving x_i, $i = 0, \ldots, \tau$ in the Laurent series expansion of $\sqrt{r}$ at ∞, i.e.

$$[\sqrt{r}]_\infty = a_\tau x^\tau + \cdots + a_0$$

and $([\sqrt{r}]_\infty)^2$ agrees with r for powers of x^k, $k = 0, \ldots, 2\tau$. A formula can be derived for the coefficients as before and matched with coefficients of powers of x in *squo*.

In step (2) of Saunders' algorithm, it is necessary to generate all possible sequences $(s_0, s_1, \ldots, s_m)$ where each $s_i \in \{0, \pm 1, \pm 2, \ldots, \pm\frac{n}{2}\}$. The quantities s_i are implemented as a one-dimensional table of rational numbers and the entire vector treated as a $m+1$ digit number base $n+1$ in order to generate all the combinations. If the vector s is initialised to

$$\begin{array}{cccc} s_m & s_{m-1} & \cdots & s_0 \\ -\frac{n}{2} & -\frac{n}{2} & & -\frac{n}{2} \end{array}$$

then repeatedly adding 1 until s is

$$\begin{array}{cccc} s_m & s_{m-1} & \cdots & s_0 \\ \frac{n}{2} & \frac{n}{2} & & \frac{n}{2} \end{array}$$

will produce all the combinations required.

As in Saunders' implementation, step 3 of the algorithm is implemented in three separate procedures for $n = 1$, $n = 2$, and $n = 4, 6, 12$. The three routines are very similar. Each generates a polynomial p of degree d as $x^d + a_{d-1}x^{d-1} + \cdots + a_1 x + a_0$ and then generates an expression as a function of p and the given θ that must be zero. By computing the numerator of the generated expression (using *normal* and *numerator*), then extracting the coefficients of each power of x, a set of expressions is found, each of which must be zero. These equations are linear functions of the unknown coefficients a_i and can be solved using a linear equation solver (courtesy of M. B. Monagan). This routine is noteworthy in that it can handle the case where the system of equations is overdetermined.

5.2. Limitations of the Implementation

While the algorithm as stated claims to solve any d.e. of the form $ay'' + by' + cy = 0$ where $a, b, c \in \mathbf{C}(x)$, the implemented algorithm will handle only a subset of these equations for a number of reasons.

In the preliminary step of the implemented algorithm, it is necessary to compute r as $\frac{s}{t}$ where $\gcd(s,t) = 1$. As long as $r \in \mathbf{Q}(x)$, Maple's *normal* will correctly simplify r to the form required. At present, however, it will not accept constants such as $\sqrt{2}$. In this case, calling radsimp will correctly normalise but radsimp will not correctly handle any more complicated constants, such as exp(2) for example.

It is clear that more powerful normalisation facilities are needed in Maple to handle the non-rational coefficients.

In the first step of the implemented algorithm, it is necessary to find the roots of a polynomial in $C[x]$. In reality, the polynomial must be in $\mathbf{Q}[x]$ for several reasons. First, the roots computed by solve are simplified using radsimp which can handle rationals and rationals to rational powers, and no other constants. Then, if solve is unable to find the roots of the polynomial, Maple's factor package is called and it can only factor over the integers (actually over the Gaussian integers if (-1)**(1/2) is represented as I).

The use of the factor package could be a limitation in and of itself in that it is very slow and relatively untested. Fortunately, for most of the examples tested (and hopefully most of the examples to be tried in the future), the polynomial we require the roots of is of low degree, generally not more than degree 4 and can be adequately handled by solve and radsimp.

Another problem with Maple's solve function manifested itself when implementing step 3 of the algorithm. In that stage, a polynomial is constructed with undetermined coefficients and an expression depending on that polynomial is set to zero. The unknown coefficients are determined by expanding the expression and extracting the coefficients of each power of the dependent variable. This often results in an over-determined system of equations. Maple's solve routine will only handle a square system of equations, i.e. n equations in n unknowns. However, M. B. Monagan was able to provide a linear equation solver that handles both under- and over- determined systems of equations as well as square systems. This routine (or a variant of it) should be provided in the Maple library.

Two problems arise in the final stages of the implemented algorithm. In step 3 in all 3 cases, the solution is returned as $e^{\int \text{something}}$. Then in the course of transforming the equation from the form $y'' = ry$ back to the input form $az'' + bz' + cz = 0$, another $e^{\int \text{something}}$ is computed. In the course of testing, the limitations of Maple's integrator were clearly demonstrated at this point. It is a well-known fact that much more work is needed on this function and I will not belabour the point.

In the cases where the integral could be computed, the second problem became evident. If the solution to the d.e. is a polynomial, it will often be computed as $e^{\log(polynomial)}$ and since Maple had no knowledge of the relationship between exp and log, it would remain in that form in the output solution. This form could hardly be considered simplified and it seemed clear that any user of the package would not appreciate such a solution. Fortunately, B. Char produced a simplifier for expressions with exp's and log's. This system of routines knows the basic rules of exponentials and logarithms, namely $e^a \cdot e^b = c^{a+b}$ and $\log x + \log y = \log xy$, and also knows that exp and log are related by $\exp(\log(\cdots)) = \cdots$. (It does not yet know that $\log(\exp(\cdots)) = \cdots$; hopefully, this can also be implemented.)

5.3. Some Further Observations

The implemented algorithm was tested extensively using the equations in Kamke [1]. During this testing another problem was noted, namely that the algorithm will not handle parameterised equations, i.e. if the input equation is $ay'' + by' + cy = 0$, then a, b and c must be polynomials in x and no other parameters. In step (2), a decision must be made as to whether the quantity d is greater than or equal to zero and is an integer. If the expression for d is parameterised and the question was simply, is $d \geq 0$, we could proceed by following the two cases, $d < 0$ and $d \geq 0$, through the remainder of the code and outputting the conditions on d along with the solutions computed in each case. However, the question, is d an integer, does not lend itself to handling a finite number of cases. A true constant must be determined for d.

In step (3) of the algorithm for case (3) only, an equation for ω is computed of degree 4, 6 or 12. It may be impossible to produce an explicit expression for one of the solutions of this equation if it is of degree 6 or 12. In that case, we return an unevaluated integral, $e^{\int \omega}$, in the solution and a condition on ω, namely the equation we were unable to solve.

These two cases demonstrate a clear need for Maple (or any other system) to be able to handle and simplify expressions with side relations. The system must have certain basic side relations built in, e.g. $i^2 + 1 = 0$, $\sin^2 x + \cos^2 x = 1$, etc. The user must also be able to add side relations to the knowledge of the system, whether directly or indirectly, e.g. via conditions on a solution to a differential equation.

In the preliminary step of the algorithm, it is necessary to perform a square-free factorisation of a polynomial. Square-free factorisation is also required by Maple's radsimp routine, (undoubtedly) by the factor package, and by the Risch integration package (as yet, not in the Maple library). It is probably also used by other routines

as well. It is recommended that a good square-free factorisation routine (for both univariate and multivariate polynomials) be added to the Maple library. The four routines mentioned above all use a slightly different square-free factorisation routine, though they may all be using the same algorithm.

The determination of the coefficients of the partial fraction expansion of a r in step (1) of the algorithm requires a routine to do polynomial long division. Maple's divide routine divides only if the input polynomials divide exactly. It is suggested that a library routine to do quotient-remainder division might be useful to a number of packages.

APPENDIX A

Source Code

```
read '/u/csmith/cs787/radsimp.m';
read '/u/csmith/essay/lsolve.m';
read '/u/bwchar/mathware/symbolic/maple/functioncall';
read '/u/bwchar/mathware/symbolic/maple/lncontract';
read '/u/bwchar/mathware/symbolic/maple/scanmap';

#
#
#--> osolve:  Solve second order ordinary differential equations
#
#     Calling sequence:  osolve(in_ode,dep,indep)
#
#     Purpose:  Solves second order ordinary differential equations
#               of the form a * y'' + b * y' + c * y = 0 using
#               Kovacic's algorithm for second order linear
#               homogeneous equations
#
#     Input:  in_ode - either an equation or an expression assumed
#                      equal to zero representing the differential
#                      equation to be solved
#
#             dep - the dependent variable to be solved for, given
#                   as an undefined function e.g. y(x)
#
#             indep - the independent variable
#
#     Output:  function value -- a set of two independent solutions
#                                of the o.d.e. if they can be found
#
#     Functions required:  odeorder,kovode
#
#

osolve := proc(in_ode,dep,indep)

  local ode,a,b,c,d,i;

  if type(in_ode,'=') then
    ode := expand(op(1,in_ode)-op(2,in_ode));
```

```
else
  ode := expand(in_ode);
fi;

if odeorder(ode,dep,indep) <> 2 then
  ERROR('not a second order o.d.e');
fi;

a := 0; b := 0; c := 0; d := 0;

if type(ode,'+') then
  for i from 1 to nops(ode) do
    op(i,ode);
    if has(",diff(diff(dep,indep),indep)) then
      a := a + " / diff(diff(dep,indep),indep);
    elif has(",diff(dep,indep)) then
      b := b + " / diff(dep,indep);
    elif has(",dep) then
      c := c + " / dep;
    else
      d := d + ";
    fi;
  od;
else
  if has(ode,diff(diff(dep,indep),indep)) then
    a := ode / diff(diff(dep,indep),indep);
  elif has(ode,diff(dep,indep)) then
    b := ode / diff(dep,indep);
  elif has(ode,dep) then
    c := ode / dep;
  else
    d := ode;
  fi;
fi;

if (indets(a) + indets(b) + indets(c)) - {indep} <> {} then
  ERROR('invalid coefficients');
fi;
```

```
  if d <> 0 then
    print('WARNING: non-homogeneous, trying homogeneous case');
  fi;

  kovode(a,b,c,indep);

end;
```

```
#
#
#--> odeorder:  Determines the order of an ordinary differential equation
#
#     Calling sequence:  odeorder(ode,dep,indep)
#
#     Purpose:  Determines the order of an o.d.e., i.e. the highest
#               derivative of the dependent variable with respect to
#               the independent variable.
#
#     Input:  ode - an equation representing the o.d.e.
#
#             dep - the dependent variable, e.g. y(x)
#
#             indep - the independent variable, e.g. x
#
#     Output:  function value -- integer order of the o.d.e., -1 if the
#                                 input equation is not an o.d.e.
#
#     Functions required:
#
#

odeorder := proc(ode,dep,indep)

  if type(ode,function) and (op(0,ode) = 'diff') then
    if op(2,ode) <> indep then
      RETURN(-1);
    elif op(1,ode) = dep then
      RETURN(1);
    else
      odeorder(op(1,ode),dep,indep);
      if " = -1 then
        RETURN(-1);
      else
        RETURN("+1);
      fi;
    fi;
  elif type(ode,'+') or type(ode,'*') then
```

```
    map(odeorder,{op(ode)},dep,indep);
    RETURN(max(op(")));
  elif type(ode,'**') then
    RETURN(odeorder(op(1,ode),dep,indep));
  else
    RETURN(-1);
  fi;

end;
```

```
#
#
#--> kovode:  Kovacic's algorithm for second order o.d.e.'s
#
#     Calling sequence:  kovode(fa,fb,fc,var)
#
#     Purpose:  Solve a second order linear homoegeneous differential
#               equation of the form fa * y'' + fb * y' + fc * y = 0
#
#     Input:  fa,fb,fc - coefficients in the differential equation,
#                     must be in Q(x)
#
#     Output:  function value -- set of two independent solutions of
#                           the o.d.e. and possibly an equation
#                           that is a condition on the solutions
#
#     Functions required:  normal,numerator,rdivide,sqfr,undetcoeff,
#                       roots,radsimp,solve,step3n1,step3n2,step3n4,
#                       esimp,int
#
#

kovode := proc(fa,fb,fc,var)

  local s,t,squo,srem,tcont,sdec,m,ord_inf,listl,oddti,i,j,k,l,
       d,theta,dfix,thetafix,ds,thetas,t1,t2,trest,rlist2,rlisthigher,
       k1,k2,soln,soln1,soln2,ac,rt,nu,vtemp,n,alls,sprod;

# transform the equation to the form z'' = (s/t)*z

  s := 2*diff(fb,var)*fa - 2*fb*diff(fa,var) + fb*fb - 4*fa*fc;
  t := 4*fa*fa;

# step 0 - preliminaries

  normal(s/t);
  s := numerator(",'t');

  rdivide(s,t,var,'squo','srem');
```

```
sdec := sqfr(t,var,'tcont');
m := nops(sdec);
t := tcont;
for i from 1 to m do
  t := t * op(i,sdec)**i;
od;
if m > 0 then
  t1 := op(1,sdec);
else
  t1 := 1;
fi;
if m > 1 then
  t2 := op(2,sdec);
else
  t2 := 1;
fi;

ord_inf := degree(t,var) - degree(s,var);

listl := [];

oddti := true;
for i from 3 by 2 to m do
  if op(i,sdec) <> 1 then
    oddti := false;
    break;
  fi;
od;

if oddti and (type(ord_inf/2,integer) or (ord_inf > 2)) then
  listl := [op(listl),1];
fi;

if not oddti or (t2 <> 1) then
  listl := [op(listl),2];
fi;

if (m <= 2) and (ord_inf >= 2) then
  listl := [op(listl),4,6,12];
```

```
  fi;

  if nops(listl) = 0 then
    FAIL();
  fi;

# step 1 - form parts for d and theta

  dfix := (min(ord_inf,2) - degree(t,var) - 3 * degree(t1,var)) / 4;
  thetafix := normal((diff(t,var) / t + 3 * diff(t1,var) / t1) / 4);

  rlist2 := roots(t2,var);

  t2 := t2 / lcoeff(t2);

  k2 := nops(rlist2);

  for i from 1 to k2 do
    trest := t / t2**2;
    rt := op(i,rlist2);
    for j from 1 to i-1 do
      trest := trest * (var - op(j,rlist2))**2;
    od;
    for j from i+1 to k2 do
      trest := trest * (var - op(j,rlist2))**2;
    od;
    undetcoeff(srem,trest,var,rt,2,2);
    d[i] := radsimp((1+4*“)**(1/2));
    theta[i] := radsimp(d[i]/(var-rt));
  od;

  k1 := k2;
  if member(1,listl) then
    for i from 4 by 2 to m do
      op(i,sdec);
      rlisthigher := roots(“,var);
      sdec := [op(1..i-1,sdec),““/lcoeff(““),op(i+1..m,sdec)];
      nu := i/2;
      for j from 1 to nops(rlisthigher) do
```

```
      k1 := k1 + 1;
      rt := op(j,rlisthigher);
      trest := t / op(i,sdec)**i;
      for l from 1 to j-1 do
        trest := trest * (var-op(l,rlisthigher))**i;
      od;
      for l from j+1 to nops(rlisthigher) do
        trest := trest * (var-op(l,rlisthigher))**i;
      od;
      undetcoeff(srem,trest,var,rt,2*nu,2*nu);
      ac[nu] := radsimp("**(1/2));
      for k from nu-1 by -1 to 2 do
        ac[k] := vtemp;
        0;
        for l from nu by -1 to k do
          " + ac[l] * ac[nu+k-l];
        od;
        ac[k] := solve("=undetcoeff(srem,trest,var,rt,2*nu,k+nu),vtemp);
      od;
      0;
      for k from 2 to nu-1 do
        " + ac[k] * ac[nu+1-k];
      od;
      d[k1] := (undetcoeff(srem,trest,var,rt,2*nu,nu+1) - ")/ac[nu];
      0;
      for k from 2 to nu do
        " + ac[k] / (var - rt)**k;
      od;
      theta[k1] := 2 * " + d[k1] / (var - rt);
    od;
  od;
fi;

if ord_inf > 2 then
  d[0] := 1;
  theta[0] := 0;
elif ord_inf = 2 then
  lcoeff(s) / lcoeff(t);
  d[0] := radsimp((1+4*")**(1/2));
```

```
    theta[0] := 0;
  elif member(1,listl) then
    nu := (-ord_inf) / 2;
    ac[nu] := radsimp(coeff(squo,var,2*nu)**(1/2));
    for i from nu-1 by -1 to 0 do
      ac[i] := vtemp;
      0;
      for j from i to nu do
        " + ac[j] * ac[i+nu-j];
      od;
      ac[i] := solve("=coeff(squo,var,i+nu),vtemp);
    od;
    0;
    for l from 0 to nu-1 do
      " + ac[l] * ac[nu-l-1];
    od;
    if nu = 0 then
      coeff(srem,var,degree(t,var)-1)/lcoeff(t) - ";
    else
      coeff(squo,var,nu-1) - ";
    fi;
    if " = 0 then
      d[0] := 0;
    else
      d[0] := " / ac[nu];
    fi;
    0;
    for l from 0 to nu do
      " + ac[l] * var**l;
    od;
    theta[0] := 2*";
  else
    d[0] := 0;
    theta[0] := 0;
  fi;

# step 2 - form trial d's and theta's

  for i from 1 to nops(listl) do
```

```
    n := op(i,listl);
    if n = 1 then
      m := k1;
    else
      m := k2;
    fi;

    if (n = 2) and (ord_inf < 2) then
      d[0] := 0;
      theta[0] := 0;
    fi;

    for j from 0 to m do
      sq[j] := -1/2 * n;
    od;

    alls := false;
    while not alls do

      sq[0] * d[0];
      for l from 1 to m do
        " - sq[l] * d[l];
      od;
      ds := radsimp(n * dfix + ");

      if type(ds,integer) and ds >= 0 then
        0;
        for l from 0 to m do
          " + sq[l] * theta[l];
        od;
        thetas := radsimp(n * thetafix + ");

# step 3 - determine polynomial P if possible and hence omega and solution

        soln := step3(n,ds,thetas,s/t,var);

        if op(1,[soln]) <> '@FAIL' then
          fb/fa;
          soln1 := exp(int(-1/2 * ",var)) * op(1,[soln]);
```

```
            soln2 := soln1 *
                  int(esimp(exp(-int("",var))/(soln1*soln1)),var);
            if nops([soln]) = 1 then
              RETURN([esimp(soln1),esimp(soln2)]);
            else
              RETURN([esimp(soln1),esimp(soln2)],op(2,[soln]));
            fi;
          fi;
        fi;

        for j from m by -1 to 0 do
          if sq[j] = (1/2 * n) then
            sq[j] := -1/2 * n;
          else
            sq[j] := sq[j] + 1;
            break;
          fi;
        od;

        if j < 0 then
          alls := true;
        fi;
      od;

    od;

    FAIL();

end;
```

```
#
#
#--> step3:  Step 3 of Kovacic's algorithm
#
#     Calling sequence:  step3(n,d,theta,rhs,var)
#
#     Purpose:  Perform step 3 of Kovacic's algorithm
#
#     Input:  n -- degree of omega over C(x)
#
#             d -- degree of the polynomial to be constructed
#
#             theta -- trial function theta
#
#             rhs -- right hand side of the o.d.e z'' = r*z
#
#             var -- independant variable of the o.d.e.
#
#     Output:  function value -- solution of the o.d.e. z'' = r*z
#                                namely exp(int(omega))
#
#     Functions required:  radsimp,numerator,Lsolve,int
#
#

step3 := proc(n,d,theta,rhs,var)

  local p,listv,i,a,pr,sete,soln,trial,w;

  p := var**d;
  listv := [];
  for i from d-1 by -1 to 0 do
    a.i := evaln(a.i);
    p := p + a.i * var**i;
    listv := [op(listv),a.i];
  od;

  pr[n] := -p;
  for i from n by -1 to 0 do
```

```
  pr[i-1] := normal(-diff(pr[i],var) - theta * pr[i]
          - (n-i) * (i+1) * rhs * pr[i+1]);
od;

trial := expand(numerator(radsimp(pr[-1])));

if trial <> 0 then
  sete := {};
  for i from ldegree(trial,var) to degree(trial,var) do
    coeff(trial,var,i);
    if " <> 0 then
      sete := sete + {"};
    fi;
  od;

  soln := Lsolve(sete,listv);
  if op(1,[soln]) = [] then
    RETURN('@FAIL');
  fi;

  for i from d-1 by -1 to 0 do
    a.i := op(2,op(d-i,soln));
  od;
fi;

trial := 0;
for i from 0 to n do
  trial := trial + pr[i] * w**i / (n-i)!;
od;

[solve(trial,w)];
if " = [] then
  RETURN(exp(int('@W'(var),var)),subs(w='@W',trial)=0);
fi;

w := radsimp(op(1,"));

exp(int(w,var));
```

```
  RETURN(“);

end;
```

```
#
#
#--> roots:  Find the roots of a polynomial
#
#     Calling sequence:  roots(poly,var)
#
#     Purpose:  Find the roots of a given polynomial in Z[var]
#
#     Input:  poly -- a univariate polynomial in var with integer
#                  coefficients
#
#           var -- the indeterminate of the polynomial
#
#     Output:  function value -- a list of the radically simplified
#                           roots of the polynomial, an ERROR
#                           if we could not find degree(poly,var)
#                           roots, i.e. all of them
#
#     Functions required:  solve,radsimp,factor
#
#

roots := proc(poly,var)

  local newp,rlist,i;

  if degree(poly,var) = 0 then
    RETURN([]);
  fi;

  soln := [solve(poly,var)];

  if nops(soln) = degree(poly,var) then
    RETURN(map(radsimp,soln));
  fi;

  newp := factor(poly);

  if newp = poly then
```

```
    ERROR('unable to find roots of the demoninator');
  fi;

  rlist := [];
  for i from 1 to nops(newp) do
    roots(op(i,newp),var);
    rlist := [op(rlist),op(")];
  od;

  RETURN(rlist);

end;
```

```
#
#
#--> sqfr:  Perform a square-free factorization of a polynomial
#
#     Calling sequence:  sqfr(poly,var,'cont')
#
#     Purpose:  Do a square-free factorization of a univariate polynomial
#
#     Input:  poly -- a univariate polynomial with integer coefficients
#
#           var -- the indeterminate of the polynomial
#
#     Output:  function value -- a list of the form [t1,t2,t3,...,tm]
#                           where poly = t1 * t2**2 * t3**3 *
#                           ... * tm**m
#
#            'cont' -- (call-by-name) if the third argument is
#                    present the integer content of poly is assigned
#                    to the name 'cont'
#
#     Functions required:  primpart,gcd
#
#     Reference:  Yun's paper "On Square-Free Decomposition Algorithms"
#
#

sqfr := proc(poly,var,cont)

  local i,signp,pp,tc,tlist1,c,d;

  signp := sign(poly);
  expand(poly) / signp;

  if nargs > 2 then
    pp := primpart(",{var},'tc');
    cont := tc * signp;
  else
    pp := primpart(",{var});
  fi;
```

```
  tlist1 := [];

  expand(diff(pp,var));
  gcd(pp,",'c','d');

  for i from 1 while c <> 1 do
    expand(d - diff(c,var));
    gcd(c,",'c','d');
    tlist1 := [op(tlist1),"];
  od;

  tlist1;

end;
```

```
#
#
#--> undetcoeff:  Determine the coefficient of a factor in a partial
#                 fraction expansion
#
#     Calling sequence:  undetcoeff(num,rden,var,root,m,ex);
#
#     Purpose:  Determine the coefficient of 1 / (var - root)**ex
#               in the partial fraction expansion of
#               num / (rden * (var - root)**m)
#
#     Input:  num -- polynomial with rational coefficients as above
#
#             rden -- polynomial with rational coefficients as above
#
#             var -- indeterminate of quotient
#
#             root -- root of the denominator of quotient
#
#             m -- multiplicity of root in denominator
#
#             ex -- particular coefficient required
#
#     Output:  function value -- coefficient as described above
#
#     Functions required:  normal, rsubs (only until 3.1 is released)
#
#     Reference:  CRC Standard Mathematical Tables
#
#

undetcoeff := proc(num,rden,var,root,m,ex)

  local k,p,i;

  k := m - ex;
  p := num / rden;

  for i from 1 to k do
```

```
    p := diff(p,var);
  od;

  p := normal(p);

  RETURN(rsubs(p,var=root)/(k!));

end;
```

```
#
#
#--> rdivide:  Divide one polynomial by another and return quotient and
#              remainder
#
#     Calling sequence:  rdivide(a,b,var,'quo','rem')
#
#     Purpose:  Divides a polynomial "a" by a polynomial "b" and produces
#               the quotient and remainder polynomials
#
#     Input:  a -- univariate polynomial with rational coefficients
#
#             b -- univariate polynomial with rational coefficients
#
#             var -- indeterminate of the two polynomials
#
#
#     Output:  function value -- none (of any relevance)
#
#              'quo' -- (call-by-name) the quotient when a is divided
#                       by b
#
#              'rem' -- (call-by-name) the remainder when a is divided
#                       by b
#
#     Functions required:
#
#     Reference:  Knuth, Volume 1, Algorithm D
#
#

rdivide := proc(a,b,var,quo,rem)

  local m,n,exa,exb,i,j,u,v,q;

  m := degree(a,var);
  n := degree(b,var);
  exa := expand(a);
  exb := expand(b);
```

```
  for i from 0 to m do
    u[i] := coeff(exa,var,i);
  od;
  for i from m+1 to n-1 do
    u[i] := 0;
  od;

  for i from 0 to n do
    v[i] := coeff(exb,var,i);
  od;

  for i from m-n by -1 to 0 do
    q[i] := u[n+i] / v[n];
    for j from n+i-1 by -1 to i do
      u[j] := u[j] - q[i] * v[j-i];
    od;
  od;

  0;
  for i from 0 to n-1 do
    " + u[i] * var**i;
  od;
  rem := ";

  0;
  for i from 0 to m-n do
    " + q[i] * var**i;
  od;
  quo := ";

end;
```

```
#
#
#--> esimp:  Simplify expressions with exponentials and logarithms
#
#     Calling sequence:  esimp(expr)
#
#     Purpose:  Simplify an expression with exp's and log's (ln's)
#               using the standard rules for exp's and log' and
#               the rule exp(log(...)) = ...
#
#     Input:  expr -- expression with exp's and log'
#
#     Output:  function value -- simplified version of expr
#
#     Functions required:  scanmap,expcontract,lncontract,explnsimp
#
#

esimp := proc(expr)

  scanmap(expr,[expcontract,lncontract,explnsimp]);
  RETURN(");

end;

save '/u/csmith/essay/kovode.m';
quit;
```

APPENDIX B

Examples and Tests

An Example of the Use of Kovacic's Algorithm

We will consider the differential equation

$$x^2z'' - 2z = 0$$

Making the transformation

$$y = z \cdot e^{-\int \frac{b}{2a}dx} = z$$

the equation becomes

$$y'' = \frac{2}{x^2}y$$

so that

$$r = \frac{2}{x^2}$$

There is only one pole of r, $c = 0$, and it has order 2, so that $\Gamma = \{2\}$. The order of r at ∞ is degree(x^2) − degree$(2) = 2$.

Checking the necessary conditions (section 2.3), we find that all three cases are possible so we must try the sub-algorithms for all three cases.

First, the algorithm for case (1). The pole $c = 0$ is of order 2 so $E_0 = \{\frac{1}{2} + k\sqrt{1+4b}\}$, $k = \pm\frac{1}{2}$, where $b = 2$ (the coefficient of $\frac{1}{x^2}$ in the partial fraction expansion of r at 0), i.e. $E_0 = \{\frac{1}{2} + \frac{1}{2}\sqrt{1+4\cdot 2}, \frac{1}{2} - \frac{1}{2}\sqrt{1+4\cdot 2}\} = \{2, -1\}$.

The order of r at ∞ is 2 so $E_\infty = \{\frac{1}{2} + k\sqrt{1+4b}\}$, $k = \pm\frac{1}{2}$, where $b = 2$ (the coefficient of $\frac{1}{x^2}$ in the Laurent series expansion of r at ∞), i.e. $E_{in} = \{\frac{1}{2} + \frac{1}{2}\sqrt{1+4\cdot 2}, \frac{1}{2} - \frac{1}{2}\sqrt{1+4\cdot 2}\} = \{2, -1\}$.

There are four possible tuples to consider; $(e_0, e_\infty) = (2,2)$, $(2,-1)$, $(-1,2)$ and $(-1,-1)$. Since $d = e_\infty - \sum_{c\in\Gamma} e_c$ and $\theta = \sum_{c\in\Gamma} \frac{e_c}{x-c}$, the possible values for d and θ are

e_0	e_∞	d	θ
2	2	0	$\frac{2}{x}$
2	-1	-3	$\frac{2}{x}$
-1	2	3	$\frac{-1}{x}$
-1	-1	0	$\frac{-1}{x}$

We can eliminate the second tuple since in that case d is not a non-negative integer.

We now search for a monic polynomial of degree d satisfying $P'' + 2\theta P' + (\theta' + \theta^2 - r)P = 0$. For the first tuple, $d = 0$ so P must be 1. We check whether this satisfies the required equation.

$$1'' + 2\cdot\frac{2}{x}\cdot 1' + \left(\left(\frac{2}{x}\right)' + \left(\frac{2}{x}\right)^2 - \frac{2}{x^2}\right) = \frac{-2}{x^2} + \frac{4}{x^2} - \frac{2}{x^2} = 0$$

So we have the correct P and θ.

Now

$$\omega = \theta + \frac{P'}{P} = \frac{2}{x} + \frac{1'}{1} = \frac{2}{x}$$

and the solution is

$$\eta = e^{\int \frac{2}{x}dx} = e^{2\log x} = x^2$$

We can transform this back using the inverse transformation

$$z = y \cdot e^{\int \frac{b}{2a}dx} = y$$

and get $z_1 = x^2$. Then by the method of reduction of order, the second solution is

$$z_2 = z_1 \cdot \int \frac{e^{-\int \frac{b}{a}dx}}{z_1^2}dx = x^2 \cdot \int \frac{1}{\left(x^2\right)^2}dx = x^2 \cdot \int \frac{1}{x^4}dx = x^2 \cdot \frac{-1/3}{x^3} = \frac{-1}{3x}$$

An Example of the Use of the Maple Implementation

The following is a listing of a Maple session using the implementation of Kovacic's algorithm.

```
Script started on Fri Aug 12 21:04:31 1983
Warning: no access to tty; thus no job control in this shell...
% maple <testrun
     |\^/|
._|\|   |/|_.
 \  MAPLE  /  Version 3.0 --- May 1983
 <____ ____>
      |
# read '/u/csmith/essay/kovode.m';
words used 1399
# prettyprint := 0;
words used 34125
prettyprint := 0
# eq1 := diff(y(x),x,x) + y(x) = 0;
eq1 := diff(diff(y(x),x),x)+y(x)=0
# osolve(eq1,y(x),x);
words used 36194
      .
      .
      .
words used 132824
[exp((-1)*I*x),1/2/I*exp(I*x)]
# eq2 := diff(y(x),x,x) + 4*x*diff(y(x),x) + (4*x**2+2)*y(x) = 0;
eq2 := diff(diff(y(x),x),x)+4*x*diff(y(x),x)+(4*x**2+2)*y(x)=0
# osolve(eq2,y(x),x);
words used 134826
      .
      .
      .
words used 147080
[exp((-1)*x**2),exp((-1)*x**2)*x]
# eq4 := x**2*diff(y(x),x,x) - 2*x*diff(y(x),x) + (x**2+2)*y(x) = 0;
eq4 := x**2*diff(diff(y(x),x),x)-2*x*diff(y(x),x)+(x**2+2)*y(x)=0
```

```
# osolve(eq4,y(x),x);
words used 149144
      .
      .
      .
words used 206842
[x*exp((-1)*I*x),1/2*x/I*exp(I*x)]
# eq5 := (x-2)**2*diff(y(x),x,x) - (x-2)*diff(y(x),x) - 3*y(x) = 0;
eq5 := (x+(-2))**2*diff(diff(y(x),x),x)-(x+(-2))*diff(y(x),x)-3*y(x)=0
# osolve(eq5,y(x),x);
words used 208881
      .
      .
      .
words used 279114
[(x+(-2))**(-3/2)*(x**2-4*x+4)**(1/4),(-8*x+6*x**2-2*x**3+1/4*x**4)*
(x+(-2))**(-3/2)*(x**2-4*x+4)**(1/4)]
# map(radsimp,");
words used 281126
      .
      .
      .
words used 418931
[(x+(-2))**(-1),(-8*x+6*x**2-2*x**3+1/4*x**4)/(x+(-2))]
# quit;
Final 'words used'=420608,  storage=1047028
%
script done on Fri Aug 12 21:12:15 1983
```

Bibliography

[1] Kamke, E., *Differentialgleichungen Losungsmethoden und Losungen*, New York, Chelsea Publishing Co., (1948).

[2] Kaplansky, I., *An Introduction to Differential Algebra*, Hermann, Paris, (1957).

[3] Kovacic, Jerald J., "An Algorithm for Solving Second Order Linear Homogeneous Differential Equations", (preprint), (1979?).

[4] Knuth, Donald E., *The Art of Computer Programming, Volume II, Seminumerical Algorithms*, Addison-Wesley, Menlo Park, California, (1973).

[5] Saunders, B. David, "An Implementation of Kovacic's Algorithm for Solving Second Order Linear Homogeneous Differential Equations", pp. 105-108 in *Proceedings of the 1981 ACM Symposium on Symbolic and Algebraic Computation*, Association for Computing Machinery, (1981).

[6] Selby, Samuel M., *CRC Standard Mathematical Tables*, CRC Press, Inc., Cleveland, Ohio, (1974).

[7] Yun, David Y. Y., "On Square-Free Decomposition Algorithms", pp. 26-35 in *Proceedings of the 1976 ACM Symposium on Symbolic and Algebraic Computation*, Association for Computing Machinery, (1976).