Solving First Order Differential Equations Using the Prelle-Singer Algorithm

R. Shtokhamer

Technical Report 88-09
Final version

May 1988

Center For Mathematical Computation
University of Delaware
Newark, DE 19716

3

Abstract

In [Prel 83] Prelle-Singer proved a structure theorem concerning elementary first-integrals for a system of autonomous first order differential equations. This theorem leads to a semidecision procedure for solving implicitly first order differential equations. We report here on a successful extension and implementation of such an algorithm in Macsyma. This is an extended writeup of previously reported results [Shto 86].

1 Introduction

Consider a two dimensional system of autonomous differential equations

$$\frac{dx}{dt} = x = P(x,y)\,,\ \frac{dy}{dt} = (x\log(\frac{x^2}{y}) + 2)y = Q(x,y). \tag{1}$$

The associate first order equation is

$$P(x,y)\frac{dy}{dx} = Q(x,y). \tag{2}$$

Define the differential operator D by

$$D\,W = P(x,y)\,\partial_x W + Q(x,y)\,\partial_y W = 0. \tag{3}$$

A function $W(x,y)$ such that $W(x(t),y(t))$ is constant on the solutions of the system (1) or equivalently $DW = 0$, is called a *first integral* for the system (1). For a first integral $W(x,y)$ it follows that $W(x,y) = c$ (for arbitrary constant c) can be considered as a general implicit solution of the differential equation (2). In our case $W(x,y) = x + \log(\log(\frac{x^2}{y}))$.

In this report we outline a generalization of the method discovered by Prelle and Singer [Prel 83] and give detailed implementation in Macsyma.

We assume that the reader is familiar with [Prel 83] and we follow the notations used therein. In the next section we formalize and generalize the approach.

2 Definitions and Generalizations

Let $\mathbf{C}$ be the complex field and $\mathcal{K} = \mathbf{C}(t_1,\cdots,t_n)$ be a finitely generated purely transcendental extension of $\mathbf{C}$. Let δ_1 and δ_2 be two derivations such that $\delta_1 x = 1$, $\delta_1 y = 0$, and $\delta_2 x = 0$, $\delta_2 y = 1$. Let $\mathcal{K}$ be a differential field with $\mathbf{C}$ being the constant field, and the differential operator D defined as $W = P\,\delta_1 + Q\,\delta_2$, where $P, Q \in \mathcal{K}$. Let $\mathcal{L}$ be a differential extension of $\mathcal{K}$. Suppose $\exists W \in \mathcal{L}$ such that

$$D\,W = P\,\delta_1\,W + Q\,\delta_2\,W = 0. \tag{4}$$

Suppose now that $t_1,\cdots,t_n$ are functions of x and y, then W can also be considered as a function of x and y. Let $y(x)$ be defined implicitly by $W(x,y(x)) = c$. Then we have:

$$\frac{dW}{dx} = \frac{dy}{dx}\delta_y(W) + \delta_x(W) = 0.$$

Using the equation (4) this can be transformed to :

$$P\,\frac{dy}{dx} = Q.$$

On the other hand not every solution of the diffferential equation will lead to a first integral. For example consider the differential equation

$$3(x^2-4)\frac{dy}{dx}+y^2-xy-3=0$$

One can check that y defined by $W(x,y) = y^4 - 6y^2 - 4xy - 3 = 0$ is a solution of the above differential equation but $W(x,y)$ is not a first integral.

The Prelle-Singer theorem leads to a decision procedure for finding W for $\mathcal{L}$ *elementary* over $\mathcal{K}$.

We will show now that the function W is directly related to an integrating factor for an associated differential form $Q\,dx - P\,dy$.

Let V_2 be a two dimensional vector space over $\mathcal{L}$. Let $v = (P,Q) \in V_2$ and $\nabla W = (\delta_x W, \delta_y W) \in V_2$. We have $\nabla W \cdot v = 0$, hence ∇W must be proportional to $(-Q,P) \in V_2$, i.e. $\exists \alpha \in \mathcal{L}$ such that $\nabla W = \alpha(-Q,P)$, i.e. $\delta_x W = -\alpha Q$ and $\delta_y W = \alpha P$, and $\delta_x(\alpha P) = \delta_y(-\alpha Q)$. Using those relations it follows that:

$$D\,W = 0 \leftrightarrow D(\alpha)/\alpha = -(\delta_x P + \delta_y Q).$$

Once the above α is found W can be calculated by quadratures:

$$A = \int^x \alpha Q dx\,, \quad W = A + \int^y (-\alpha P - \delta_y(A))\,dy. \tag{5}$$

3 Adoption of the Prelle-Singer Theorem

Assume now that $\mathcal{L}$ is an *elementary* extension of $\mathcal{G}$. Let the constant fields of $\mathcal{L}$ and $\mathcal{G}$ (of the derivations δ_x and δ_y) be equal. Let D be defined as before, and let I be a non-trivial element of $\mathcal{L}$ i.e. $\delta_x(I) \neq 0$ or $\delta_y(I) \neq 0$ then:

Theorem 1 (Prelle-Singer) *: There exist elements $w_0, w_1, \cdots, w_m$ of $\mathcal{L}$* **algebraic** *over $\mathcal{G}$ and constants $c_1, \cdots, c_m$ such that:*

$$D(w_0) + \sum_{i=1}^{m} c_i\, D(w_i)/w_i = 0 \quad \text{and} \quad \delta(w_0) + \sum_{i=1}^{m} c_i\, \delta(w_i)/w_i \neq 0$$

for $\delta = \delta_x$ or $\delta = \delta_y$.

What is important to observe is that if there exist an elementary first integral then there exists one of the form $I = w_0 + \sum_{i=1}^{m} c_i \log(w_i)$. Hence the derivatives $\delta_x(I)$ and $\delta_y(I)$ are algebraic over $\mathcal{G}$. But $\alpha\, P = \delta_y(I)$ and $\alpha\, Q = -\delta_y(I)$ (Not both of P and Q being zero), i.e. α is algebraic over $\mathcal{G}$ as well. Recall that $D(\alpha)/\alpha = -(\delta_x(P) + \delta_y(Q))$. P and Q are in the differential field $\mathcal{G}$, so is $D(\alpha)/\alpha$. This is enough to show that actually α must be a *radical* over $\mathcal{G}$ [Prel 83]. Therefore $\exists \beta \in \mathcal{G}$ and an integer m such that $\alpha^m = \beta$ or $\beta^{1/m} = \alpha$, and

$$1/m\, D(\beta)/\beta = -(\delta_x(P) + \delta_y(Q)).$$

In $\mathcal{G}$ we may write

$$\delta_x = \sum_{i=1}^{n} X_i \partial_{t_i} \;\; \delta_y = \sum_{i=1}^{n} Y_i \partial_{t_i}.$$

Where $X_i = \delta_x(t_i)$ and $Y_i = \delta_y(t_i)$ are elements in $\mathcal{G}$. Using the above relation, writing all expressions as ratios of relatively prime polynomials in $\mathcal{K}[t_1, \cdots, t_n]$, multiplying by the least common multiple lcm of all the denominators in the equation for β we arrive finally at :

$$1/m\, \tilde{D}(\beta)/\beta = S. \tag{6}$$

Where $\tilde{D} = \sum_{i=1}^{n} P_i\, \partial_{t_i}$, $P_i = (P\, X_i + Q\, Y_i)\, lcm$, P_i's and S are elements in $\mathcal{K}[t_1, \cdots, t_n]$, i.e. polynomials.

deg, (β, . . .)

4 Adoption of the Prelle-Singer Algorithm

Let $\beta = \prod_{i=1}^{l} f_i^{n_i}$, where the n_i's are non zero integers, and the f_i's are irreducible polynomials in $\mathcal{K}[t_1, \cdots, t_n]$ then it follows from equation (6):

$$\sum_{i=1}^{l} r_i \tilde{D}(f_i)/f_i = S. \tag{7}$$

Where the rational numbers r_i are given by $r_i = n_i/m$. Since the f_i's are relatively prime polynomials and S is a polynomial, we may conclude (by partial fraction decomposition argument) that f_i divides $\tilde{D}(f_i)$, i.e. $\forall i,\ 1 \leq i \leq l\ \exists g_i \in \mathcal{K}[t_1, \cdots, t_n]$ such that :

$$\tilde{D}(f_i) = f_i\, g_i. \tag{8}$$

Solving the equation (8) is the major step in the Prelle-Singer algorithm. To proceed we must introduce a total order on the monomials in $\mathcal{K}[t_1, \cdots, t_n]$. Let $\mathcal{M}$ be the set of all monomials (including 1), considered as a monoid under multiplication. Let Φ be a bijection from the set of positive integers onto the set $\mathcal{M}$. We will say that Φ introduces an *admissible* order $\succ$ on the set $\mathcal{M}$ if and only if:

1. $\Phi(1) = 1$
2. $\Phi(i) \succ \Phi(j) \leftrightarrow i > j$
3. $i > j \rightarrow\ \forall u \in \mathcal{M}\ (u\,\Phi(i) \succ u\,\Phi(j))$

For $A \in \mathcal{K}[t_1, \cdots, t_n]$ let $M(A)$ be the maximal monomial in A and let the degree *deg* of A be defined by $deg(A) = \Phi^{-1}(M(A))$. Notice that $deg(1) = 1$.

It is clear that bound on the degree of the polynomials g_i in equation (8) can be found independent on the f_i's. For example we may take as a bound the number n_d defined by $n_d = max_{i=1}^{n}(deg(P_i))$. Without loss of generality we may assume that the f_i's are monic, i.e. the coefficient of the highest monomial is unity. Suppose we found n_s different solutions f_i of equation (8).

Lemma 1 *If $n_s > n_d$ then there exist constants $c_i \in \mathcal{K}$ and a function $I \equiv \prod_{i=1}^{n_s} f_i^{c_i}$ such that $D(I) = 0$.*

Proof: The g_i's can be viewed as elements in a vector space over $\mathcal{K}$ spanned by the monomials up to degree n_d. Hence if $n_s > n_d$ the g_i's from equation (8) must be linearly dependent, i.e. $\exists c_i \in \mathcal{K}$, $1 \leq i \leq n_s$ and $\sum_{i=1}^{n_s} c_i g_i = 0$ or $\tilde{D}(\prod_{i=1}^{n_s} f_i^{c_i}) = 0$. But $\tilde{D}(F) = 0 \leftrightarrow D(F) = 0$.

Notice that if the c_i are non rationals the above first integral I not elementary, and I is not a rational function in $\mathcal{G}$, however we still call such a "simple" solution "rational". From the lemma we know that we need to seek for at most $n_d + 1$ f_i's solving the equation (8). The problem is to find the solutions.

We make now the following assumption, which is an extension of the theorem by Darboux as quoted from [Jouan 79] in [Prel 83].

Assumption 1 *There exists an integer n_f such that if h is irreducible and $h \mid \tilde{D}(h)$ then $deg(h) \leq n_f$.*

The last assumption makes the Prelle-Singer algorithm a semidecision procedure and not a *bone fide* decision procedure. The reason being that we do not know how to compute the number n_f.

Having a bound n_f we may easily transform the equation $\tilde{D}(h) = h\,g$ into set of $\Phi^{-1}(\Phi(n_d)\,\Phi(n_f))$ equations (some of them might be trivial) in n_d x $(n_f - 1)$ unknowns. The solutions of these equation (algebraic variety) are assumed to be in the field $\mathcal{K}$. So the field $\mathcal{K}$ is defined the to be the minimal extension of $\mathcal{R}$ including the above mentioned algebraic variety.

Once the solutions are obtained and $n_s > n_f$ we have a "rational" solution mentioned in the lemma. Otherwise we must still solve the equation (7). It may happen that we can solve the equation (7) only for $r_i \in \mathcal{K}$ i.e. not necessarily rational numbers, in that case the integrating factor obtained will not in general lead to an elementary integral.

Not having the bound n_f our implementation seeks f_i's of progressively higher degree till some *a priori* bound, set by the user is met.

5 Implementation of the Algorithm

Following the discussion in the previous section the algorithm may be summarized as follows:

1. Solve equation (8) .
   SF $\leftarrow \emptyset$
   For m=1 thru bound-on-f do (
   S1 $\leftarrow$ all irreducible monic polynomials f of degree m such that $f \mid \tilde{D}(f)$
   SF $\leftarrow$ S1 $\cup$ SF)

2. Decide if there are constants (or rational numbers) c_f such that
   $\sum_{f \in SF} c_f \tilde{D}(f)/f = 0$. If such constants are found return a "rational" solution.

3. Solve equation (7).
   Decide if there exist constants c_f such that $\sum_{f \in SF} c_f \tilde{D}(f)/f = S$, in such a case construct the integrating factor $\alpha = \prod_{f \in SF} f^{c_f}$, and compute a first integral using integrations as in equation 3. 05

4. Report failure if SF is empty or step 2 and step 3 failed to produce a result.

- **Step 1.**
  Let the bijection Φ be given. (In our implementation we use the total degree ordering). Let $f = \Phi(m) + \sum_{i<m} f_i \Phi(i)$ and $g = \sum_{i=1}^{n_d} g_i \Phi(i)$. Define N to be such that $\Phi(N) = \Phi(m)\Phi(n_d)$. Recall that we seek f and g such that $\tilde{D}(f) = fg$ We view the polynomial g as an n_d dimensional vector $\vec{g}$, and the polynomial f in fg as linear operator represented in this case as a Nxn_d matrix F. $\tilde{D}(f)$ is considered as an N dimensional vector $\vec{h}$. Let the elements of the matrix F be denoted by $F_{i,j}$ then:

$$F_{i,j} = f_k, \text{ where } \Phi(i) = \Phi(k)\Phi(j).$$

We solve the equations in $F\vec{g} = \vec{h}$ as follows. Consider the following n_d rows of the matrix F

$$\Phi^{-1}(\Phi(1)\Phi(m)),\ \Phi^{-1}(\Phi(2)\Phi(m)), \cdots, \Phi^{-1}(\Phi(n_d)\Phi(m))$$

Consider now the equations emerging using these rows only. Let $\hat{F}$ be the corresponding n_dxn_d matrix i.e. whose j'th row is the $\Phi^{-1}(\Phi(j)\Phi(m))$ row of the original matrix F. Let $\hat{h}$ be the corresponding vector of $\vec{h}$. We claim that $\hat{F}$ is an upper diagonal matrix with 1's on the diagonal. By the definition of $\hat{F}$ we have $\hat{F}_{l,j} = F_{i,j}$, where $i = \Phi^{-1}(\Phi(l)\Phi(m))$, and $1 \le l, j \le n_d$. Therefore $\Phi(i) = \Phi(l)\Phi(m)$, but $F_{i,j} = f_k$ and $\Phi(i) = \Phi(k)\Phi(j)$ i.e. $\Phi(l)\Phi(m) = \Phi(k)\Phi(j)$. But $k \le m \rightarrow \Phi(k) \preceq \Phi(m) \rightarrow \Phi(l) \preceq \Phi(j)$, i.e. $l \le j$. Therefore $\hat{F}$ is upper diagonal. For $\hat{F}_{j,j}$ we have in the previous identities $j = l$ therefore $m = k$ and $\hat{F}_{j,j} = f_m = 1$.

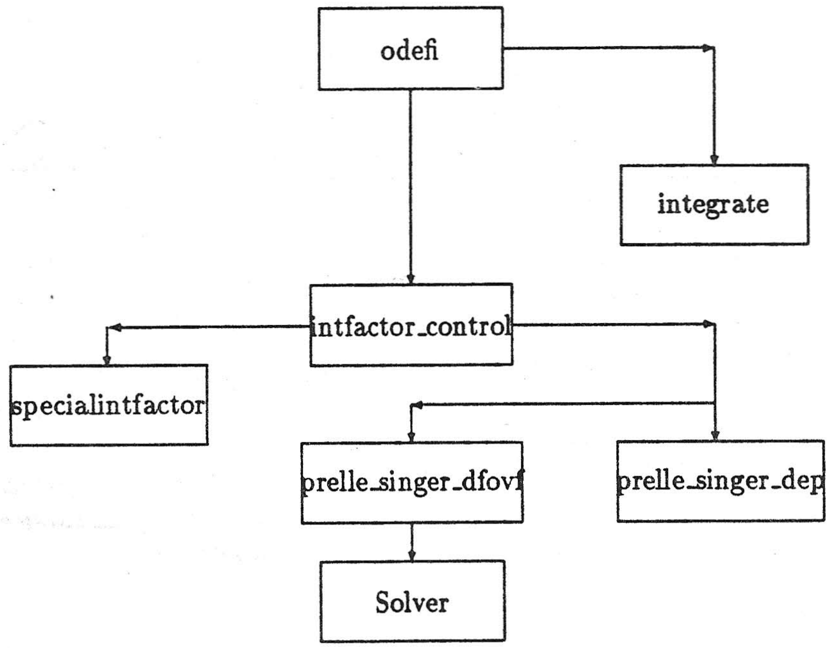
Therefore the linear equation $\hat{F}\vec{g} = \hat{h}$ for the n_d unknowns in $\vec{g}$ can be solved by simple substitutions and the solutions will be polynomials in the entries of $\hat{F}$ and $\hat{h}$, i.e. polynomials if the unknowns f_i's. Let eliminate from the equation $F\vec{g} = \vec{h}$ all of the above treated ones. In this way we get $N - n_d$ non-linear equations for the $m-1$ unknowns $f_1, f_2, \cdots, f_{m-1}$. This system of non-linear equation is solved using available algebraic solver. Our implementation uses the algebraic solver of MACSYMA. For generality it is required that the solver should be able to solve properly equations over algebraic extensions, and return all of the solutions together with the defining polynomial for the primitive element of the new algebraic field.

- **Steps 2 and 3**
If constants in $\mathcal{K}$ are sought then consider the polynomials g obtained in the previous step as vectors $\vec{g}$ as explained before.
If rational constants are needed then let $q_1, \cdots, q_l$ be indeterminate constants (the ones defining the field $\mathcal{R}$ as finite extension of the rational field $\mathbf{Q}$) and let γ be a primitive element defining the algebraic extension $\mathcal{K}$ say of degree k over $\mathbf{Q}_l=\mathbf{Q}(q_1, \cdots q_l)$. Consider now the vectors $\vec{g}$ as elements in a vector space spanned freely over $\mathbf{Q}$ by the monomials $q_1^{n_1}..q_l^{n_l}\gamma^j$, where $0 \le j < k$.
Let t be the number of solutions obtained in step 1, and set $g_{t+1} = S$. Find (by some elimination-diagonalization procedure) if $\vec{g}_j$ is linearly dependent on $\vec{g}_1, \cdots, \vec{g}_{j-1}$. If such j is found and $j \le t$ then there exists a "rational" first integral. If $j = t+1$ then the integrating factor α, is computed and the first integral is obtained by integration.

In the above it was assumed that the polynomials f are dependent on some transcendentals $t_1, \cdots, t_n$. Assume now that $t_1, \cdots, t_n$ are algebraically dependent. If a "rational" solution is found (i.e. in step 2) the rational function might be trivial. For let $F(t_1, \cdots, t_n) = 0$. (such a polynomial F exists because $t_1, \cdots, t_n$ are algebraically dependent). Clearly $D(F) = 0$ but F is not a (non-trivial) first integral. Moreover we may find two polynomials $F1, F2$ and $F1 \ne F2$ such that $I = F1(t_1, \cdots, t_n)/F2(t_1, \cdots, t_n) = 1$ Factoring $F1 = \prod_i f1_i^{n_i}$, $F2 = \prod_j f2_j^{m_j}$ (assuming for a moment that $t_1, \cdots, t_n$ are algebraically independent), we get $\sum_i n_i D(f1_i)/f1_i - \sum_j m_j D(f2_j)/f2_j = 0$ However I is trivial i.e. $\delta_x I = \delta_y I = 0$. So some solutions obtained in step 2 must be rejected if they are trivial. The integrating factor obtained in step 3 will lead to a nontrivial first integral.

It follows from the above discussion that we can extend the algorithm to treat algebraically dependent variables. It should be clear however that such a heuristic approach is incomplete. Solutions in steps 2 or 3 might not be found (assuming the t_i's being algebraically independent) which might exits if the algebraic dependence would be taken into account.

6 The Main Flow-Chart



7 Description

- odefi :
  The top level call to the new differential equation solver.
  The syntax is the same as in the call to **ode** i.e. `odefi(eq,y,x)` where `eq` is a differential equation, y the dependent variable and x the independent one.

- integrate :
  Integrating routine of Macsyma. Needed to compute a first integral from equation 2.

- intfactor_control :
  The controlling routine of the new package. Determines according to the flag **type_of_intfactor** if a special and/or the Prelle_Singer integrating factor will be computed. Checks if the transcendentals form a closed differential field.

- specialintfactor :
  Special type of integrating factors for some classes of differential equations taken from the book " Introduction to Non-Linear Differential and integral equations", by H. Davis, Dover Publications Inc. 1962.

- prelle_singer_dfovf :
  Procedure solving the **Df/f** problem. The maximal degree of **f** is determined by the input parameter **degree,** as determined form the global variable **maxterms.** If maxterms < 0 then abs(maxterms) is used otherwise max("number of transcendentals"+1,maxterms) is used for the degree (see the procedure intfactor_control). This is done to insure in the regular case that **f** will be at least linear in the transcendentals.
  Further control can be obtained by the use of the parameter **nterms** determined from the global **listofterms** in "intfactor_control". If **listofterms** is not false then if it is a positive integer it will determine the maximal number of terms that the polynomial **f** can have. If it is a list than only the terms explicitly listed in the sublists of that list will be tried for the solution for **f**.

- Solver :
  Small module solving lower diagonal linear equations and non-linear equations over algebraic extension fields. The current implementation uses only the **algsys** procedure of MACSYMA.

- prelle_singer_dep :
  This procedure determines the linear dependency as explained in the steps 2 and 3. The current implementation uses the Gauss elimination method. The type of the solution is controlled by the two flags :**elementary** and **primitiveflag.** To seek an integrating factor leading to an elementary solution the flag **primitiveflag** can be set to **true** and the flag **elementary** set to 1.

Flags and Global Variables:

METHOD_FI : The method used
INTFACTOR_FI : The corresponding integrating factor.

In case of failure METHOD_FI may contain information suggesting resetting some global variables to enhance the chances of success.(See below)

There are several flags controlling the execution path:

- type_of_intfactor : Possible settings are :

  1 : Only the standard methods will be tried.

  2 : Only the P-S algorithm will be tried.

  3 : (default) Both of methods will be attempted.

- elementary : Possible settings (for the P-S algorithm) are :

  1 : Use only an integrating factor which is a radical of a rational function, or return a "simple" solution i.e. a solution for which there is no need for an integrating factor.

  2 : Return (if exists) only a "simple" solution.

  3 : Return only a "simple" radical solution.

  4 : (default) Return any one of the above solutions.

- primitiveflag : Possible settings are :

  true : The P-S algorithm for finding linear dependence will work over the vector space defined by the monomials and the primitive element of the extension field.

  false : (default) Work over the vector space defined by the monomials only. In this case canonical forms are assumed for expressions in the extension field.

- MAXTERMS : This variable defines an upper bound on the degree of monomials for the polynomials p in solving the $p \mid Dp$ problem. As there is no known theoretical bound on this degree we allow this heuristic bound to be set by the user.
  Possible settings are:

  positive integer : Up to max(MAXTERMS,length(vrlst)+1) terms can be used. The default is +3,'. (vrlst is the list of the transcendentals in the equations).

  negative integer : Up to (-MAXTERMS) terms can be used.

- LISTOFTERMS : This is a variable enabling restrictive search for solutions for the polynomials p in the $p \mid Dp$ problem.
  Possible settings are:

  false : (default) All terms up to the degree determined by MAXTERMS will be used.

  positive integer : Only this number of terms will by tried.

  A list of lists of terms : Only those terms appearing in the lists will be tried in solving the $p \mid Dp$ problem.

- ODEFI_GRAD_LIST : This variable if not [] allows substitution of new variables for subexpressions in the differential equations. Especially it is useful in reducing integer high powers of variables. This option should be used with type_of_intfactor=2. If used the implicit derivatives of the new variables should be declared using the gradef function.
  The possible settings are:

  [ ]: default.

  list : List of the form $[var_1 = expression_1, var_2 = expression_2, ...]$.

8 Examples

In this section we demonstrate the usage of the new package by solving some typical examples mostly from Kamkes's book. We start with a differential equation which is of the generic form

$$y' * x * (a + \alpha x^m y^n) = y * (b + \beta x^m y^n)$$

for which the integrating factor is of the form $x^t y^s$.

```
(c1) eq:'diff(y,x)*(2*x^2*y-x)=2*x*y^2+y;                 /* Kamke 261 */

                            2        dy          2
(d1)                    (2 x  y - x) -- = 2 x y  + y
                                     dx

(c2) odefi(eq,y,x);

                     2 x y log(y) - 2 x log(x) y + 1
(d2)               - ------------------------------- = %c
                                   x y

(c3) method_fi;

(d3)                          DAVIS method #5, Goursat
```

The above solution was returned by the procedure *specialintfactor*. The standard differential equation solver **ode** returns a result in a slightly different form.

```
(c4) ode(eq,y,x);

                                   2 x y log(x y) + 1
                                   ------------------
                                         4 x y
(d4)                      x = %c %e

(c5) method;

(d5)                                genhom
```

The next example is one with which **ode** has difficulty, and should in that case return the result **false**. It returns however very complicated implicit result containing the derivatives of the dependent variable y.(a bug ?)

```
(c6) eq:'diff(y,x)*(y^2+x^2+x)=y;                         /* Kamke 275 */

                                2    2       dy
(d6)                          (y  + x  + x) -- = y
                                             dx
```

```
(c7) odefi(eq,y,x);

                  %i (log(y + %i x) - log(%i x - y)) + 2 y
(d7)            - ---------------------------------------- = %c
                                     2
(c8) method_fi;

(d8)                    Prelle-Singer Integrating Factor

(c9) intfactor_fi;

                                   1
(d9)                     ---------------------
                         (y - %i x) (y + %i x)
```

Notice that the result (d7) is in the *Prelle-Singer* form. If the integrating factor is transformed to the real form $1/(y^2 + x^2)$ the solution becomes $\arctan(x/y) - y = c$.

```
(c10) ode(eq,y,x);

Transform to remove non-integral exponents

(d10) y =
                 2                     2       2
                d y       dy         2 d y   2 d y           dy 2
              x (--- - 2 --) + y  --- + x  --- - 2 y (--)
            /     2       dx          2        2             dx
            [   dx                   dx      dx
            I ------------------------------------------------------ dx
            ]                    2 dy     2 dy       dy
            /                  y -- + x  -- + x --
                                 dx        dx       dx
        %k1 %e
  - ---------------------------------------------------------------
               2                     2       2
              d y       dy         2 d y   2 d y           dy 2
            x (--- - 2 --) + y  --- + x  --- - 2 y (--)
          /     2       dx          2        2             dx
          [   dx                   dx      dx
          I ------------------------------------------- dx
          ]                    2 dy     2 dy       dy
      /   /                  y -- + x  -- + x --
      [                        dx        dx       dx
  %k1 I %e                                                        dx + %k2
      ]
      /
```

In the next example the elementary methods requires a complicated integrating factor, while the Prelle-Singer method finds a solution **without** integration.

```
(c11) eq:'diff(y,x)*(y^2+2*x*y-x^2)=y^2-2*x*y-x^2;      /* Kamke 281 */
```

(d11) $$(y^2 + 2\,x\,y - x^2)\,\frac{dy}{dx} = y^2 - 2\,x\,y - x^2$$

```
(c12) odefi(eq,y,x);
```

(d12) $$\log(y^2 + x^2) - \log(y + x) = \%c$$

```
(c13) method_fi;
```

(d13) Homogeneous of order 2

The above solution was therefore obtained by the *simpleintfactor.*

```
(c14) intfactor_fi;
```

(d14) $$\frac{1}{x\,(y^2 - 2\,x\,y - x^2) + y\,(-\,y^2 - 2\,x\,y + x^2)}$$

Now for the **ode** solution.

```
(c15) ode(eq,y,x);
```

(d15) $$-\,\frac{y + x}{2\,y^2 + 2\,x^2} = \%c$$

```
(c16) method;
```

(d16) exact

```
(c17) intfactor;
```

(d17) $$\frac{1}{(y^2 + 2\,x\,y - x^2)^2 + (-\,y^2 + 2\,x\,y + x^2)^2}$$

The information given by **ode** that the equation is exact is inaccurate. **ode** used the fact that for the above example $Py' = Q$, $P(x,y) + iQ(x,y)$ is an analytic function in which case the integrating factor is known to be of the form $1/(P^2 + Q^2)$.

To enforce the Prelle-Singer algorithm we set the flag **type_of_intfactor** to 2.

```
(c18) type_of_intfactor:2;

(d18)                              2

(c19) odefi(eq,y,x);

                          (y - %i x) (y + %i x)
(d19)                     --------------------- = %c
                                  y + x
```

$\frac{y^2+x^2}{y+x} = \%c$

```
(c20) method_fi;

(d20)                  Prelle-Singer Rational Solution
```

In the next example **defi** returns an "elementary" solution while **ode** is unable to solve it.

```
(c22) eq:'diff(y,x)*x=(x*log(x^2/y)+2)*y;       /* Kamke 120 */

                                          2
                             dy          x
(d22)                      x -- = (x log(--) + 2) y
                             dx          y

(c23) odefi(eq,y,x);

                                   2
                                  x
(d23)                     log(log(--)) + x = %c
                                  y
```

$+ \log(2)$

```
(c24) method_fi;

(d24)                  Prelle-Singer Integrating Factor

(c25) intfactor_fi;

                                       1
(d25)                             -----------
                                        2
                                       x
                                  x log(--) y
                                       y
```

```
(c26) ode(eq,y,x);

(d26)                                false
```

In the next example the Prelle-Singer algorithm correctly finds that the equation is exact, i.e. the integrating factor is 1. Clearly the solution should be obtained by immediate integration.

```
(c27) eq:'diff(y,x)*2*x^3*y=-3*x^2*y^2-7;

                              3   dy          2  2
(d27)                      2 x  y -- = - 3 x  y  - 7          exact
                                  dx

(c28) odefi(eq,y,x);

                                3  2
(d28)                        - x  y  - 7 x = %c

(c29) method_fi;

(d29)                   Prelle-Singer Integrating Factor

(c30) intfactor_fi;

(d30)                                  1
```

Resetting the flag **type_of_intfactor** to 3 will allow **defi** to examine first the elementary methods incorporated in *simpleintfactor.*

```
(c31) type_of_intfactor:3;

(d31)                                  3

(c32) odefi(eq,y,x);

                                3  2
(d32)                        - x  y  - 7 x = %c

(c33) method_fi;

(d33)                                exact
```

The last example illustrates that an "rational" result can be obtained with non-rational exponents. Clearly an elementary solution exists by evaluating the logarithm of the result.

```
(c34) assume(a>0)$

(c35) eq:'diff(y,x)*(x^2+1)=a*(y^2+1);
```

(d35) $$(x^2 + 1) \frac{dy}{dx} = a (y^2 + 1)$$

```
(c36) odefi(eq,y,x);
```

(d36) $$\frac{(x - \%i) (y + \%i)^{1/a}}{(x + \%i) (y - \%i)^{1/a}} = \%c$$

```
(c37) method_fi;
```

(d37) Prelle-Singer Rational Solution

To get the elementary result we set the flag **elementary** to 1.

```
(c38) elementary:1;
```

(d38) $$1$$

```
(c39) odefi(eq,y,x);
```

(d39) $$\frac{\log(y + \%i)}{a} - \frac{\log(y - \%i)}{a} - \log(x + \%i) + \log(x - \%i) = \%c$$

odefi can treat algebraic extensions as well. Those cases however are treated hauristicaly, i.e. all the algebraic's are treated as independent transcendentals. Doing so we may not discover first integrals which may could be obtained if the algebraic dependencies were taken into acount. Also trivial (i.e. constant first integrals) obtained are rejected.

```
(c40) eq:(x*sqrt(x^2+y^2+1)-y*(x^2+y^2))*'diff(y,x)
        -y*sqrt(y^2+x^2+1)-x*(y^2+x^2) = 0;
```

(d40) $$(x \operatorname{sqrt}(y^2 + x^2 + 1) - y (y^2 + x^2)) \frac{dy}{dx} - y \operatorname{sqrt}(y^2 + x^2 + 1) - x (y^2 + x^2) = 0$$

```
(c41) radcan(odefi(eq,y,x));
```

(d41) $$\operatorname{sqrt}(y^2 + x^2 + 1) + \operatorname{atan}\left(\frac{x}{y}\right) = \%c$$

The last example demonstrates the substantial speedup which can be achieved using the **odef_grad_list**. For example to solve the equation

x^{100}

$$(2y - x^{100} + 3x)\frac{dy}{dx} = -(300x^{99} - 1)y - 299 * x^1 00$$

will require about 5000 terms. However introducing a new variable t, we may rewrite this equation as:

```
(c41) eq:'diff(y,x)*(2*y+3*x-t) = -(300*t/x-1)*y-299*t;

                                      dy       300 t
(d41)               (2 y + 3 x - t) -- = - (----- - 1) y - 299 t
                                      dx         x

(c42) ODEFI_GRAD_LIST:[t = x^100]$

(c43) gradef(t,x,100*t/x)$

(c44) type_of_intfactor:2$

(c45) odefi(eq,y,x);

                     3       100  2        200        300
                    y  + 3 x    y  + 3 x     y + x
(d45)               -------------------------------- = %c
                                 y + x
```

Acknowledgment

The author wishes to thank Prof. B. Caviness for introducing the problem and for his help in finding a partial solution.

References

[Prel 83] M. J. Prelle, M. F. Singer *Elementary First Integrals of Differential Equations*, Trans. Amer. Math. Soc.,C vol. 279, **1**, 1983.

[Shto 86] R. Shtokhamer, B. F. Caviness, N. Glinos *Computing Elementary First Integrals of Differential Equations*, "Computers and Mathematics" , Conference, Stanford , Aug. 1986.

[Jouan 79] J. P. Jouanolou *Equations de Pfaff algèbraiques*, Lectures Notes in Math., vol. 708, Springer-Verlag Berlin and New York, 1979