MACSYMA PROGRAM FOR THE PAINLEVE TEST OF NONLINEAR
ORDINARY AND PARTIAL DIFFERENTIAL EQUATIONS
WISCONSIN UNIVERSITY
CENTER FOR MATHEMATICAL SCIENCES
W. HERMAN ET AL.

UNCLASSIFIED

DEC 87 CMS-TSR-88-21 AFOSR-85-0263

F/G 12/4

NL
CMS Technical Summary Report #88-21
MACSYMA PROGRAM FOR THE PAINLEVÉ TEST
OF NONLINEAR ORDINARY AND PARTIAL
DIFFERENTIAL EQUATIONS
Willy Hereman and Eric Van den Bulck

Center for the Mathematical Sciences
University of Wisconsin—Madison
610 Walnut Street
Madison, Wisconsin 53705

December 1987

(Received December 15, 1987)

Approved for public release
Distribution unlimited

Sponsored by
Air Force Office of Scientific Research
Rolling Air Force Base
Washington, DC 20332
UNIVERSITY OF WISCONSIN-MADISON  
CENTER FOR THE MATHEMATICAL SCIENCES  

MACSYMA PROGRAM FOR THE PAINLEVÉ TEST OF NONLINEAR  
ORDINARY AND PARTIAL DIFFERENTIAL EQUATIONS  

Willy Hereman* and Eric Van den Bulck**  

Technical Summary Report #88-21  
December 1987  

ABSTRACT  

A MACSYMA program is presented which determines whether a given single  
nonlinear ODE or PDE with real polynomial terms fulfills the necessary  
conditions for having the Painlevé property. The program listing is given,  
together with a synopsis of the algorithm, a model of the output of the  
program, a list of tested examples and instructions to use the package.  

AMS (MOS) Subject Classifications: 34E15, 35G20, 68Q40  
PACS Classifications: 2.30Jr, 2.90, 3.20g, 3.40k  
Key Words: Painlevé property, integrability, macsyma, symbolic programming;  
soliton theory, dynamical systems.  

*Mathematics Department, University of Wisconsin-Madison, Madison, WI 53706.  
**Department of Mechanical Engineering, Solar Energy Laboratory, University of  
Wisconsin-Madison, Madison, WI 53706.  

This work was partially supported by the Air Force Office of Scientific  
Research under Grant No. 85-NM-0263.
MACSYMA PROGRAM FOR THE PAINLEVE TEST OF NONLINEAR
ORDINARY AND PARTIAL DIFFERENTIAL EQUATIONS

Willy Hereman†
and
Eric Van den Bulck‡

1. INTRODUCTION

A simple equation or system is said to have the Painlevé property, PP, if its only singularities in the complex plane consist of movable poles\(^1\). This requires that the solution \(f\) of a PDE, say in two independent variables \((t,x)\) can be expressed as

\[
f = g^\alpha \sum_{k=0}^{\infty} u_k g^k, \tag{1}
\]

with \(u_0(t,x) \neq 0\), \(\alpha\) a negative integer, and where \(u_k(t,x)\) are analytic functions in a neighbourhood of the singular, non-characteristic manifold \(g(t,x) = 0\), with \(g_x(t,x) \neq 0\).

Performing the Painlevé test, as proposed by Weiss\(^2\), involves three steps:

(i) Determination of the negative integer \(\alpha\) and \(u_0\) from the leading order ansatz \(f \propto u_0 g^\alpha\), by balancing the minimal power terms;
(ii) Identification of the non-negative integer powers \(r\), the resonances, at which arbitrary functions \(u_r\) can enter the expansion \(f \propto u_0 g^\alpha + u_r g^{\alpha + r}\). This is achieved by requiring that \(u_r\) is arbitrary after substitution of this form for \(f\) into the equation, only retaining the most singular terms;
(iii) Verification that the correct number of arbitrary functions \(u_r\) indeed exists by substituting a truncated expansion of the form \((1), k = 1, 2, \ldots, r_{\text{max}}\), where \(r_{\text{max}}\) represents the largest resonance, into the given equation. At non-resonance

† Mathematics Department and Center for the Mathematical Sciences, University of Wisconsin-Madison, Wisconsin 53706, USA
‡ Department of Mechanical Engineering, Solar Energy Laboratory, University of Wisconsin-Madison, Wisconsin 53706, USA. Future address: Departement Werktuigkunde, Katholieke Universiteit Leuven, B-3030 Heverlee, Belgium
* Sponsored by the Air Force Office of Scientific Research under Grant No. 85-NM-0263
levels, \( u_k \) should be unambiguously determined; at resonance levels \( u_j \) should be arbitrary due to a vanishing coefficient of \( g^{-m_{\text{powy}}} \) (compatibility condition). Here, \( m_{\text{powy}} \) denotes the (negative) power in \( g \) of the most singular terms in the equation.

An equation or system for which the above steps can be carried out consistently, is said to have the PP and is conjectured to be integrable\(^1\). Counter examples\(^7-10\) of integrable PDEs without the PP disprove the necessity of the PP for integrability (i.e., being exactly solvable by a linear integral equation of Inverse Scattering Transform); whereas some non-integrable equations seem to have the PP as defined by Weiss et al\(^3\), hence questioning the conjecture\(^11\) in its present form. Refinements of the PP have been established\(^12-15\), allowing for rational power expansions of \( f \), which lead to algebraic branch points in addition to movable poles.

Nevertheless, the PP has been a useful criterion for complete integrability tests\(^5,6,10-18\). It is a handy tool in the derivation of Bäcklund transformations and Lax pair representations for various PDEs\(^5,18-20\), and it gives insight in Hirota's direct method for solitary wave construction\(^21,22\), and other expansion methods which lead to rational solutions\(^18\).

2. SCOPE AND LIMITATIONS OF THE PROGRAM

The program is largely based upon the MACSYMA routine \textit{ODEPAINLEVE} developed by Rand and Winternitz\(^23\), which checks the PP for a single ODE with real polynomial terms.

The package works for both single ODEs and PDEs, with arbitrary degree of nonlinearity. In principal, the number of variables is unlimited. Although the released version of the program works with only four independent variables. PDEs may have time and space dependent coefficients of integer degree. Transcendental terms in the equation must be removed by a suitable exponential transformation of the dependent variable\(^17,24,25\).

Due to 'intermediate expression swell' it may occur that the program can not perform the test in a reasonable time. Therefore, verifying the resonances and compatibility conditions are made optional through the boolean variable \textit{do\_resonances}. The highest level of verification is controllable by entering a value for \textit{max\_resonance}. For the investigation of Bäcklund transformations one may consider calculations beyond the automatically determined level \textit{rmax} by taking \textit{max\_resonance} \( \geq \textit{rmax} \).

Intermediate output can be obtained by including extra print statements in the program. (Warning: After any alternation in the main program a new LISP version needs to be created by \textit{batch(main.p)} in MACSYMA).
Major difficulties arise when \( u_0 \) can not be substituted since it occurs in subsequent calculations at a lower degree than the one present at its evaluation. In that case, calculations by hand or in interactive MACSYMA are recommended. The vital steps for that are easily extracted from the program listing.

At present, the program does not perform the weakest forms of the Painlevé test\(^{12-15}\) although \( \alpha \) in (1) may be rational and can be positive, through a user supplied value for \( \beta \). Later extensions\(^{26}\) will cover complex equations and coupled systems. Some of these must be treated by rather general rational power expansions\(^{12-15}\). A further refinement of the algorithm, performing selective substitutions of \( u_k \) and their derivatives in the recursive relations, will allow to construct Bäcklund transformations for PDEs\(^{8,19,25}\).

3. HOW TO USE THE PROGRAM: EXAMPLES

The equation to be tested should be saved in a batch file, before entering MACSYMA.

**Example 1**: An ODE, where the use of independent variable \( x \) is mandatory.

/* Batch file R.W.(5.6) for testing Eq. (5.6) in paper Rand-Winternitz\(^{23}\) */

\[
\text{eq: } (f^2)\cdot f_x^3(x) - 3\cdot(f_x^1(x))^3$
\]

\[
\text{beta: } -2$
\]

\[
\text{max.resonance: } 5$
\]

/* beta and max.resonance are specified here */

Note that, during the test the variable \( x \) will be replaced by \( g = x - x_0 \), where \( x_0 \) is the arbitrary initial value of \( x \).

**Example 2**: A PDE, with independent variables \( t, x, \) and \( y \).

/* Batch file KP, for the Kadomtsev-Petviashvili Equation\(^{3}\) */

\[
\text{eq: } ftxyx[1,1,0][t,x,y] + (ftxy[0,1,0][t,x,y])^2 + f^*ftxy[0,2,0][t,x,y] + b*ftxy[0,4,0][t,x,y] + ftxy[0,0,2][t,x,y]$
\]

/* b is a constant; beta, do resonances and max resonance are not specified */

Similarly, using \( ftxy[x,y,z] \), evolution and wave equations in \( t, x, y \) and \( z \) can be entered. For parameterical equations the explicit dependence on the variables must be given (e.g. coefficients \( a(t), b(x), \ldots \)). The five first letters of the alphabet are reserved to denote arbitrary constants, if necessary to be supplemented with \( a1, a2, \ldots, b1, \ldots \).
Once the batch files are made, enter MACSYMA and subsequently type:

(c1) batch(main.p) $ → to create painleve.lsp, only to be done ones
(c2) batch(setup.p) $ → to repeat before every new example
(c3) batch(kp) $ → reads in the batch file kp
(c5) writefile(test.kp) $ → opens output file test kp
(c6) batch(exec.p) $ → starts the actual Painleve test
(c7) closefile() $ → closes output file test kp

4. TEST RUNS

Although the program has been successfully tested, with eunice MACSYMA version 309.3 on a VAX 11/780, for more than 30 examples including all ODEs and selected PDEs from the cited references, unexpected situations may still occur. In case of trouble, the printout gives vital information to remedy the problems. The program does not rely on non-standard MACSYMA routines (e.g. powers is provided).

Model of Output:

For the Space Dependent Burgers equation$^{10,20} : -x^2f_{2x} - xff_x + f_t = 0$, which passes the test:

\[
\begin{align*}
\text{PAINLEVE ANALYSIS OF EQUATION, } & - x^2 f_{2x} - x f_f + f_x = 0 \\
\text{SUBSTITUTE } u g & \text{ FOR } f \text{ IN ORIGINAL EQUATION.} \\
\text{MINIMUM POWERS OF } g \text{ ARE } & [2 \alpha - 1, \alpha - 2] \\
\text{• COEFFICIENT OF } g & = \quad 2 \alpha - 1 \\
\text{• COEFFICIENT OF } g & = \quad 2 \alpha - 2 \\
\text{FOR EXPONENTS (2 } \alpha - 1 \text{ AND (} \alpha - 2 \text{) OF } g \text{, DO} & \\
\text{WITH } \alpha & = -1, \text{ POWER OF } g \text{ IS -3 } \rightarrow \text{ SOLVE FOR } u \\
\text{TERM } & = u g x (2 g x - u) \quad \text{IS DOMINANT IN EQUATION.} \\
\text{WITH } u & = 2 g x \rightarrow \text{ FIND RESONANCES} \\
\text{SUBSTITUTE } u g & = u g + u g \quad \text{FOR } f \text{ IN EQUATION}
\end{align*}
\]
TERM \(- (g^2 \frac{r-2}{r+1} x^u g^r - 3)\) IS DOMINANT IN EQUATION.

THE ONLY NON-NEGATIVE INTEGRAL ROOT IS \([r = 2]\)

WITH MAXIMUM RESONANCE = 2 \(\rightarrow\) CHECK RESONANCES.

\[
\begin{array}{c}
\text{SUBSTITUTE POWER SERIES} \\
\sum_{k=0}^{g-1} g^k u^k \\
\end{array}
\]

WITH \(u = 2 g x\)

COEFFICIENT OF \(1\) IS \(2 g x (g x + u g x - g)\)

\[
\frac{2}{g}
\]

\(u = g x - g\)

COEFFICIENT OF \(1\) IS 0

\(u\) IS ARBITRARY

COMPATIBILITY CONDITION IS SATISFIED

The results of test runs on 15 other examples are summarised in Table 1.

5. LISTING

5.1 Listing of setup.p routine

```c
/* The batch file SETUP_P, which initializes the program after every example */
kill (all)$
loadprint : false$
loadfile ("painleve.lsp")$
derivabbrev : true$
depends([(f).eq, [t,x,y,z]])$
fx[k](x):=diff(f,x,k)$
ftx[k,l](t,x):=diff(diff(f,t,k),x,l)$
fty[k,l,m](t,x,y):=diff(diff(diff(f,t,k),x,l),y,m)$
ftxyz[k,l,m,n](t,x,y,z):=diff(diff(diff(diff(f,t,k),x,l),y,m),z,n)$
```

5.2 Listing of exec.p routine

```c
/* The batch file EXEC_P, to execute the test for an example */
exec_painleve (eq, beta, do_resonances, max_resonance)$
```
5.3 Listing of main.p routine

/* The batch file MAIN.P, with the main program for the Painleve test */
define_variable (do_resonances, true, boolean)$
declare (max_resonance, integer)$
nonposInt (expres) :- if integerp(expres) then is(expres<0) else false$
solveforhipow (expr, var) :- block([m],
expr : expand(expr),
m : hipow(expr, var),
return(part(solve(expr, var m), 1)))$
konstant (expr, var, q) :- block([h, co],
if q # 1 then (  
  co : ratcoef(expand(expr), var, 0),
  co : ratcoef(co, var, 0)
),
return (co))$
pow(expr, var) :- block(
[power_list : [1], ff : expand(expr), tmp, tmp1, infun, inflag : true],
declare(power_list, special),
if not atom(ff) and part(ff, 0) = "+" then infun : args(ff) else infun : [ff],
for i in influn do {
  tmp1 : i1%,
  if not atom(tmp1) and part(tmp1, 0) = "-" then tmp1 : substpart("+", tmp1, 0),
  if atom(tmp1) or part(tmp1, 0) = "+" then tmp1 : [tmp1],
  tmp : maplist(lambda([u], get_exponent(u, var), tmp1), tmp1),
  tmp : delete([1], tmp),
  if tmp = "[]" then (  
    if not member(0, power_list) then power_list : endcons(0, power_list)
  )
  else (  
    if not member(first(tmp), power_list) then
      power_list : endcons(first(tmp), power_list)
  )
),
power_list)$
get_exponent(expr, var) :- block(
if freeof(var, expr) then return('[]),
if atom(expr) then return(1),
if inpart(expr, 0) = "+" and inpart(expr, 1) = var then
return(inpart(expr, 2))
else
return('[]))$
check_resonances(eq, valalfa, minpow, uzero, rmax, s) :- block(
[series, eq1, eq2, dalfa.co.uk],
print (" WITH MAXIMUM RESONANCE = ", rmax, "  " -> CHECK RESONANCES."),
print (" SUBSTITUTE POWER SERIES ", *sum(u[k]*gt(valalfa+k), k, 0, rmax),
" FOR f IN EQUATION.").
series : sum(u[k]*gt(valalfa+k), k, 0, rmax),
eq1 : ratsubst(series, f, eq),
eq1 : ev(eq1, diff),
eq1 : expand(eq1 / (gt(minpow))),
eq1 : ratsimp(ratsubst(0, gt(s*rmax+1), eq1)),
co : denom(eq1),
if integerp(co) and co # 1 then eq1 : num(eq1),
if uzero # [] then eq1 : ratsubst(rhs(expand(uzero)), lhs(uzero), eq1),
dalfa : denom(abs(valalpha)).
co : konstant (eq1,g,dalfa).
if co # 0 then 
  print("*** FATAL ERROR : NON-ZERO LEADING COEFFICIENT REMAINS ",
  "AFTER SUBSTITUTING ", uzero, ", ***)
  return ()
). if dalfa # 1 then eq1 : radcan(eq1).
if uzero # [] then print("WITH", uzero)
else print("WITH", u[0], "ARBITRARY I")
for i : 1 thru sermax do block ([i],
  if dalfa # 1 then for j : 1 thru dalfa-1 do 
    co : ratcoef(expand(eq1),g,j/dalfa),
    if co # 0 then 
      if not freeof(g,co) then 
        print("*** FATAL ERROR : COEFFICIENT STILL CONTAINS g. **")
        return 
      ). /* end if */
    eq1 : radcan(eq1 - co*gt(j/dalfa))
  )

eq1 : ratsimp(eq1/g).
co : konstant (eq1,g,dalfa),
eq1 : eq1 - co,
co : ratsimp(xthru(co)),
if not integerp(co) then co : factor(co),
print("COEFFICIENT OF",g*(l+minpowg)."IS ",co),
k : i/s,
if co # 0 and integerp(k) then 
  if ratcoef(co,u[k],s) = 0 then 
    if not freeof(u[k],co) then 
      print("*** RESONANCE VIOLATED BY APPEARANCE OF",u[k],
      "FROM NON-LEADING TERMS. **")
    else block([top,j,cox0],
      print(" ", u[k]." IS ARBITRARY!").
      print("COMPATIBILITY CONDITION: ", co. " = 0").
      if (ratsimp(ev(co.dif)) = 0) then 
        print("CONDITION IS SATISFIED")
      else 
        print("*** CONDITION NOT SATISFIED. **")
    print("*** CHECK FOR FREE PARAMETERS OR PRESENCE OF",
      u[0]." ". **")

  top : num(co),
  if not freeof(x0,top) then for j;0 while top # 0 do 
    top : expand(top),
    cox0 : coeff(top,x0,0),
    top : (top-cox0)/x0,
    cox0 : factor(cox0),
    print("COEFFICIENT OF ",x0j." IN NUMERATOR IS ",cox0)
  )
) /* end do */
else block ([temp],
  uk : solve(co,u[k].s),
  uk : part(uk.i),
  temp : factor(ev(rhs(uk).dif)),
  uk : ratsubs(temp,rhs(uk),uk),
  print(" ",uk),
  eq1 : ratsubs(rhs(uk),lhs(uk),eq1),
  if uzero # [] then eq1 : ratsubs(rhs(expand(uzero)),lhs(uzero),eq1),
  eq1 : ev(eq1, dif)
) /* end else if */
else if co = 0 and integerp(k) then (
print ("", u[k]ts, "IS ARBITRARY!").
print ("COMPATIBILITY CONDITION IS SATISFIED!")
)
*/ end of function check_resonances */

find_resonances(eq, valalfa, minpowg, uzero) :- block (}
[eql, eq2, coefr, a, i, l, temp1, temp2, powU0, dalfa],
print ("WITH", uzero, "-> FIND RESONANCES").
print ("SUBSTITUTE", u[0]*gtalfa, "+", u[r], gt(alfa+r),
"FOR f IN EQUATION").
eq1 : ratsubst(u[0]*(gtvalalfa)+u[r]*(gt(r+valalfa)), f, eq).
eq1 : ev(eq1, diff).
eq1 : ratsimp(eq1/(gtminpowg)).
dalfa : denom (abs(valalfa)).
if dalfa # 1 then eql : radcon(eq1).
powU0 : hipow(lhs(uzero), u[0]).
temp1 : denom(eq1), if integerp(temp1) and temp1 # 1 then eql : num(eq1),
temp2 : hipow(subst(r=1, eq1), g).
coefr : 0.
for i : 1 thru temp2 while coefr = 0 do (}
eq2 : ratcoef(eq1, u[r], i).
eq2 : ratcoef(eq2, g, i*r).
coefr : konstant (eq2, q, dalfa),
if uzero # [1 then (}
coefr : expand(coefr),

while hipow(coefr, u[0]) >= powU0 do
coefr : expand(ratsubst(rhs(expand(uzero)), lhs(uzero), coefr))
),
),
coefr : factor(num(ratsimp(radcon(coefr/u[r]ts)))),
print ("TERM", coefr.[u[r]ts, gt(r+minpowg)], "IS DOMINANT IN EQUATION.").
coefr : xthru(coefr/factor(gcd(coefr, diff(coefr,r), r))).
if part(coefr,0) = "-" then coefr : = coefr,
if part(coefr,0) = "/" then coefr : num(coefr),
temp2 : 1.
if part(coefr,0) = "+" then (}
if hipow(coefr, r) = 1 then (}
if (coeff(coefr, r) = 1 and nonposint(subst(r=0, coefr))) then

temp2 : coefr
)
else if part(coefr,0) = "*" then

for ii : 1 thru length(coefr) do block ( [pp],
pp : part(coefr, ii),
if hipow(pp, r) = 1 then (}
if coeff(pp, r) = 1 and nonposint(subst(r=0, pp)) then

temp2 : temp2 * pp
)
),
if temp2 = 1 then
print ("THERE ARE NO NON-NEGATIVE INTEGRAL ROOTS FOR r.")
else (}
temp2 : solve(temp2, r),
if length(temp2) = 1 then
print ("THE ONLY NON-NEGATIVE INTEGRAL ROOT IS ", temp2)
else print ("THE", length(temp2), "NON-NEGATIVE INTEGRAL ROOTS ARE ", temp2),
print (".")
temp2 : apply(max.map(rhs, temp2)),
if temp2 > 0 then (}
if do_resonances then (}
if integerp(max_resonance) then temp2 : max_resonance,
check_resonances(eq,valalpha,mlnpowg,uzero,tempp2,
)
else
print(" EXIT FIND_RESONANCES WITHOUT CHECKING.")
));$ */ end of function find_resonances */

find_uzero(eq,valalpha,mlnpowg) := block([eq2,lowest,uzero,dalpha],
print(" WITH alpha = ",valalpha," POWER OF g is ",mlnpowg,
" SOLVE FOR ",uzero));
eq2 := ratsubst(u(0)*gtvalalpha,f,eq),
eq2 := ev(eq2,diff),
eq2 := ratsimp(eq2 / (gtm'lnpowg)),
dalpha := denom(abs(valalpha)),
if dalpha # 1 then eq2 := radcan(eq2),
lowest := denom(eq2),
if integerp(lowest) and lowest # 1 then lowest := num(eq2),
lowest := factor(constant(eq2,g,dalpha)),
print(" TERM ",lowest,gtm'lnpowg," IS DOMINANT IN EQUATION."),
if lowest = 0 then find_resonances(eq,valalpha,mlnpowg,[])
else if freeof(u[6],lowest) then
print("*** FAILURE OF PAINLEVE TEST -- NONZERO LEADING ORDER IS ",
"FREE OF",u[6]," ABANDON THIS alpha VALUE. ***")
else (for i:1 while remainder(lowest,u[0]) = 0 do lowest := xthru(lowest/u[0]),
if freeof(u[0],lowest) then
print(" *** FAILURE OF PAINLEVE TEST. ",u[0]=0,
" ABANDON THIS alpha VALUE. ***")
else (if (denom(lowest) # 1) then lowest := num(lowest),
lowest := factor(lowest/gcd(lowest,diff(lowest,u[0]),u[0]))),
if part(lowest,0) = "-" then lowest := -lowest,
if part(lowest,0) = "/" then lowest := num(lowest),
if part(lowest,0) # "" then 
uzero := solveforhipow(lowest,u[0]),
uzero := ratsubst(factor(rhs(uzero)),rhs(uzero),uzero),
find_resonances(eq,valalpha,mlnpowg,uzero)
)
else for i:1 thru length(lowest) do 
(uzero := part(lowest,i),
if not freeof(u[i],uzero) then (uzero := solveforhipow(uzero,u[0]),
find_resonances(eq,valalpha,mlnpowg,uzero))
)
));$ */ end of function find_uzero */

paineve (eq,beta,do,resonances,max_resonance) := block(
[eql, u, alfa, f, exp, list, len, eq2, alfa1st],
print(" PAINLEVE ANALYSIS OF EQUATION. ",eq,":=0"),
print(" SUBSTITUTE "",u[0]+gtalfa," FOR f IN ORIGINAL EQUATION."),
eq1 := ratsubst(u[0]+gtalfa,f,eq),
eq1 := ev(eq1,diff),
eq1 := map(factor,eq1),
i := hipow(expand(denom(eq1)),g),
list := pow(num(eq1),g),
list := map(lambda(entry,entry-1),list),
list := apply(min,list),
if part(list,0) = min then list : args(list) else list : [list],
len := length(list),
print(" MINIMUM POWERS OF g ARE ",list),
9
exp : expand(eq1),
for i:1 thru len do (
  eq2 : expand(rotcoef(eq1,g,part(list.i))).
  eq2 : rotcoef(eq2,g,0),
  eq2 : ratsubst(0,gvalfa,eq2),
  eq2 : factor(eq2),
  print("  COEFFICIENT OF ",gpart(list.i), " IS ", eq2)
).

alfalist:[].
if len > I and not ratnump (beta) then
  for i: 1 thru len-1 do for j:i+1 thru len do
    block([
      valalfa:subst(valalfa,part(list.i),),
      valalfa : rhs(part(valalfa.i)),
      if valalfa < 0 then (list1 : subst(alfa=valalfa,list1),
        minpowg : part(list1,i),
        if not member(valalfa,nifal list) then (print("  FOR EXPONENTS ",part(list.i),") AND (",part(list.j),") OF g, DO"),
        alfalist : cons(valalfa,alfalist),
        if apply(min.listi) # minpowg then
          print("  POWER OF g IS NOT MINIMAL — SKIP THIS VALUE OF ALFA")
        else find_uzero(eq.valalfa,minpowg)
      )
    )
  )
else print("  FOR EXPONENTS (",part(list,i),") AND (",part(list.j),") OF g, "
    alfa = ", valalfa, " ALREADY DONE!"
  )
else print("  FOR EXPONENTS (",part(list.i),") AND (",part(list.j),") OF g, alfa = ", valalfa, " IS NON-NEGATIVE."
).
/* This is the end of the first line including the block */

list : subst(beta,alfa,list),
list : apply(min.list),
if ratnump(beta) then (print("  RESULTS FOR USER-SUPPLIED VALUE alfa =", beta),
  find_uzero(eq,beta,list)
)
)
/* end of function painleve */

exec_painleve (eq, beta, do.resonances, max.resonance) := block ( [
deegdfx,deegdfy,deegdfz],
deegdfx : derivdegree(eq,f.x),
deegdfy : derivdegree(eq,f.y),
deegdfz : derivdegree(eq,f.z),
if (deegdfx=0) and (deegdfy=0) and (deegdfz=0) then block([
  remove([f.g,eq],dependency),
  depends(f.g),
  depends(g.x),
  depends(u,k),
  declare(g.mainvar),
  eq1 : ev(eq,diff),
  eq1 : ratsubst(1,diff(g.x),eq1),
  deegx : derivdegree(eq1,g.x),
  for i:2 thru deegx do (eq1 : ratsubst(0,diff(g.x,ev(l)),eq1),
    eq1 : num(xthru(eq1))
  )
eq : subst(g+x0.x eql),
    print ('SUBSTITUTE', x, '->', g+x0)
)
else (
    depends([g],[t,x,y,z]),
    depends([u],[k,t,x,y,z])
)

painedev (eq, beta, do_resonances, max_resonance))$

save (["painedev.lsp"], do_resonances, functions);

ACKNOWLEDGEMENTS

This work is partially supported by AFOSR under Grant 85-NM-0263. We thank Jonathan Len (Symbolics Inc.) for writing the POWERS subroutine.
<table>
<thead>
<tr>
<th>ODE or PDE</th>
<th>REF</th>
<th>ALPHA</th>
<th>( u_0 )</th>
<th>( r \geq 0 )</th>
<th>( u_1 \leq 1 )</th>
<th>COMP</th>
<th>PP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Korteweg-deVries ( f_t + ff_x + b f_{3x} = 0 )</td>
<td>3</td>
<td>-2</td>
<td>( u_0 = -12b g_x )</td>
<td>4, 6</td>
<td>( u_1, u_3, u_5 ) det ( u_4, u_6 ) arb</td>
<td>satis</td>
<td>yes</td>
</tr>
<tr>
<td>Sine-Gordon ( 2 f f_{tx} - 2 f f_x - f^3 f = 0 )</td>
<td>27</td>
<td>-2</td>
<td>( u_0 = 4 g_t g_x )</td>
<td>2</td>
<td>( u_1 ) det ( u_2 ) arb</td>
<td>satis</td>
<td>yes</td>
</tr>
<tr>
<td>Kadomtsev-Petviashvili or twc-dimensional KdV ( f - f_{x}^{2} f_{xx} + b f_{4x} + f = 0 )</td>
<td>3</td>
<td>-2</td>
<td>( u_0 = -12 b g_x )</td>
<td>4, 5, 6</td>
<td>( u_1, u_2, u_3 ) det ( u_4, u_5, u_6 ) arb</td>
<td>satis</td>
<td>yes</td>
</tr>
<tr>
<td>Harry Dym ( f_t - f^3 f_{3x} = 0 )</td>
<td>8</td>
<td>( 2 )</td>
<td>( u_0 = \frac{3 g t}{4 g_x} )</td>
<td>( \frac{3}{3} )</td>
<td>( \frac{2}{3} )</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Bullough-Dodd ( f f_{xt} - f f_x + a f - b f = 0 )</td>
<td>19</td>
<td>-1</td>
<td>( u_0 = \frac{g_x g_t}{b} )</td>
<td>2</td>
<td>( u_1 ) det ( u_2 ) arb</td>
<td>par</td>
<td>dep</td>
</tr>
<tr>
<td>( f_t b f_{x} + f_{5x} = 0 )</td>
<td>28</td>
<td>-2</td>
<td>( u_0 = \frac{60 g_x^2}{b} )</td>
<td>2, 6, 13, 60</td>
<td>( u_1, u_3, u_4, u_5 ) det ( u_2, u_6 ) arb</td>
<td>satis</td>
<td>und</td>
</tr>
<tr>
<td>Parametrical sinh-Gordon ( f f_{tx} - f f_{x} b(t) f^3 f ) + ( a(t) (f f_{x} x f f_{2x} - x f^2 f) = 0 )</td>
<td>15</td>
<td>-2</td>
<td>( u_0 = \frac{4 g_x}{b(t)} )</td>
<td>2</td>
<td>( u_1 ) det ( u_2 ) arb</td>
<td>satis</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Table 1** Test Cases
<table>
<thead>
<tr>
<th>ODE or PDE</th>
<th>REF</th>
<th>ALPHA</th>
<th>$u_0$</th>
<th>$r \geq 0$</th>
<th>$u_{i \geq 1}$</th>
<th>COMP</th>
<th>PP</th>
</tr>
</thead>
<tbody>
<tr>
<td>$af_x + \frac{bf^3}{x} = 0$</td>
<td>23</td>
<td>-1</td>
<td>$u_0^2 = -2/b$</td>
<td>4</td>
<td>$u_1, u_2, u_3$ det $u_4$ arb</td>
<td>par dep</td>
<td>und</td>
</tr>
<tr>
<td>$(f_x + a f_x x + b x f_x x) = 0$</td>
<td>23</td>
<td>-1 $\frac{2 - 3 \sqrt{2} i}{3 + 2 i}$</td>
<td>$u_0 = 2/a$</td>
<td>2</td>
<td>$u_1^2$ det $u_2$ arb</td>
<td>satis</td>
<td>yes</td>
</tr>
<tr>
<td>$f^2 + 3f^3 x = 0$</td>
<td>23</td>
<td>$\beta = -2$</td>
<td>$u_0$ arb</td>
<td>0, 10</td>
<td>$u_1 = 0$, $1 \leq i \leq 9$ $u_{10}$ arb</td>
<td>satis</td>
<td>yes</td>
</tr>
<tr>
<td>Emden type</td>
<td>29</td>
<td>-1</td>
<td>$u_0 = 1$</td>
<td>1</td>
<td>$u_1$ arb</td>
<td>satis</td>
<td>yes</td>
</tr>
<tr>
<td>$f_x + 3f_x x + f^3 = 0$</td>
<td>29</td>
<td>-1</td>
<td>$u_0 = 2$</td>
<td>none</td>
<td>$u_1$ arb</td>
<td>satis</td>
<td>yes</td>
</tr>
<tr>
<td>Ince type</td>
<td>5</td>
<td>-1</td>
<td>$u_0 = 2$</td>
<td>2</td>
<td>$u_1 = 0$ $u_2$ arb</td>
<td>not satis</td>
<td>no</td>
</tr>
<tr>
<td>Painleve II</td>
<td>30</td>
<td>-1</td>
<td>$u_0 = \pm 1$</td>
<td>4</td>
<td>$u_1, u_2, u_3$ det $u_4$ arb</td>
<td>satis</td>
<td>yes</td>
</tr>
<tr>
<td>$f_{xx} - 2f^3 x - a = 0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Double sine-Gordon type</td>
<td>27</td>
<td>-1</td>
<td>$u_0 = \pm 1$</td>
<td>2</td>
<td>$u_1$ det, $u_2$ arb</td>
<td>satis</td>
<td>yes</td>
</tr>
<tr>
<td>$f_x - f_x^2 a (f^3 - f_x + 1 - f^4) = 0$</td>
<td></td>
<td>1</td>
<td>$u_0 = \pm 1$</td>
<td>2</td>
<td>$u_1$ det, $u_2$ arb</td>
<td>satis</td>
<td>yes</td>
</tr>
<tr>
<td>Diffusion</td>
<td>16</td>
<td>$\frac{-1}{2}$</td>
<td>$u_0 = g_x^2/(2g_t)$</td>
<td>$\frac{1}{2}$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1 cont.
6. REFERENCES

ENI DATED FILM 8-88 Drisc