Reprint from
The *Mathematica* Conference

June, 1992   Boston, MA

940.9.10.1992

Solving Nonlinear Differential Equations with DSolve

Alexei V. Bocharov

1 Abstract

Algorithms for solving nonlinear ordinary differential equations are now implemented within the 'Calculus' library extending *Mathematica*'s `DSolve` command.

The primary new feature related to the nonlinear ODE solver is the extensive use of **implicit** functions which in particular frequently leads to final results written in implicit or parametric form.

This new feature sometimes may disappoint a user but in many cases it proves to be an advantage because it allows users to work with functions that otherwise would be inaccessible. In a number of cases it also turns out that the implicit representation is much more concise and therefore much more understandable than the explicit one.

The class of nonlinear ordinary differential equations now handled by the `DSolve` is outlined here. Also, the general policy for output representation in the nonlinear part of `DSolve` is explained in more detail below and characteristic examples are given.

The `DSolve` command has been improved significantly since last presented on the Boston `Mathematica` conference in May 1992. The improvement has been both in terms of enhancing the solving power of the algorithm as well as in terms of providing for a better representation of the results.

2 First Order Nonlinear ODEs: What to Expect

The general method used for a first order differential equation of the form

$$F[x, y[x], y'[x]] == 0$$

is to find its first integral of the form:

$$I[x, y[x]] == C[1]$$

(where C[1] is the integration constant) and then to solve for the $y[x]$.

This is done using *Mathematica*'s `Solve` function.

Let us consider the following four examples characteristic of what may happen here.

■ Example 1 (Kamke 1.12).

This loads the package.

```
In[1]:= <<"/usr4/alexei/dos/CopyOfDSolve.m"
```

This is an example of a first integral nicely solved to a conventional form.

```
In[2]:= y'[x] == 1 - y[x]^2

                       2
Out[2]= y'[x] == 1 - y[x]
```

```
In[3]:= DSolve[%,y[x],x]

                                           2
DBG::DInputStatus {{y'[x] == 1 - y[x] }, {},
  {y[x]}, {}, {1}}
                                2
Out[3]= {{y[x] -> -1 + ---------------}}
                           -2 x + C[1]
                        1 - E
```

■ Example 2 (Kamke 1.38).

```
In[4]:= y'[x] == a y[x]^3 + b x^(-3/2)

                   b            3
Out[4]= y'[x] == ---- + a y[x]
                  3/2
                 x
```

This is the first integral you get in quadratures.

```
In[5]:= DSolve[%,y[x],x]

                              b            3
DBG::DInputStatus {{y'[x] == ---- + a y[x] }, {},
                             3/2
                            x

  {y[x]}, {}, {1}}
Out[5]= Solve[Integrate[
                        1
       -------------------------------------, y[x]] -
         2 b                          3
       ------- + y[x] + 2 a x y[x]
       Sqrt[x]
    Log[Sqrt[x]] == C[1], y[x]]
```

Mathematica had problems in computing it for generic a and b. (Reducing it to elliptic functions does not help much in terms of making $y[x]$ explicit.)

This however may be well simplified afterwards if the user chooses to specialize a and/or b.

For example,

```
In[6]:= %/.b:>0

Out[6]= Solve[-Log[Sqrt[x]] + Log[y[x]] -
                           2
    Log[1 + 2 a x y[x] ]
    --------------------- == C[1], y[x]]
             2
```

This seems unfortunate, but the reason this seems undersimplified is that all the functions are assumed to have complex values and the high-school level rules for logarithms do not work in a complex domain.

The user has the option to override this in the following way:

```
In[7]:= %/.{(a_. Log[b_]+c_. Log[d_]):>Log[b^a d^c]}

                       y[x]
Out[7]= Solve[Log[-------] -
                     Sqrt[x]
                           2
    Log[1 + 2 a x y[x] ]
    --------------------- == C[1], y[x]]
             2
```

```
In[8]:= %/.{(a_. Log[b_]+c_. Log[d_]):>Log[b^a d^c]}

                     C[1]
                    E     Sqrt[x]
Out[8]= {{y[x] -> ─────────────────────────},
                            2 C[1]  2
                  Sqrt[1 - 2 a E       x ]

                C[1]
               E     Sqrt[x]
  {y[x] -> -(─────────────────────────)}}
                         2 C[1]  2
              Sqrt[1 - 2 a E       x ]

In[9]:= Solve[%,y[x]]

Solve::elist:
   List encountered during logical expansion of
              C[1]
             E     Sqrt[x]
    {y[x] -> ──────────────}.

             Sqrt[1 + <<1>>]

                           C[1]
                          E     Sqrt[x]
Out[9]= Solve[{{y[x] -> ─────────────────────────},
                                  2 C[1]  2
                        Sqrt[1 - 2 a E       x ]

                 C[1]
                E     Sqrt[x]
  {y[x] -> -(─────────────────────────)}}, y[x]]
                          2 C[1]  2
              Sqrt[1 - 2 a E       x ]
```

■ Example 3 (Inverted linear equation).

```
In[10]:= y'[x] == 1/(x y[x]+1)

                       1
Out[10]= y'[x] == ──────────
                  1 + x y[x]

In[11]:= DSolve[%,y[x],x]

                                          1
DBG::DInputStatus {{y'[x] == ──────────}, {},
                                     1 + x y[x]

  {y[x]}, {}, {1}}
Out[11]= {{x[y] ->

      2            2
     y /2         y /2      Pi          y
    E     C[1] + E     Sqrt[──] Erf[──────]}}
                            2       Sqrt[2]
```

This is another instructive example : expressing the general solution as $y[x] \to$ *something* would be a hopeless task here, whereas the same general solution (understood as the set of the integral curves on the (x, y) plane is well described by $x[y] \to$ *an explicit formula.*

The user has the option of applying the `InverseFunction` here should $y[x]$ expressed via the `InverseFunction` be preferred.

■ Example 4 (Kamke 1.117).

```
In[12]:= y'[x] == Exp[y[x]/x] + y[x]/x + 1

                          y[x]/x   y[x]
Out[12]= y'[x] == 1 + E         + ────
                                    x
```

```
In[13]:= DSolve[%,y[x],x]

                                       y[x]/x   y[x]
DBG::DInputStatus {{y'[x] == 1 + E          + ----},
                                                 x

  {}, {y[x]}, {}, {1}}

Solve::ifun:
   Warning: Inverse functions are being used by
     Solve, so some solutions may not be found.

Solve::ifun:
   Warning: Inverse functions are being used by
     Solve, so some solutions may not be found.

                              y[x]/x        y[x]
Out[13]= Solve[Log[(1 + E          ) x] - ---- == C[1],
                                             x

  y[x]]
```

All the integration is successfully done in this case but the remaining first integral is clearly a transcendental equation both with respect to $y[x]$ **and** with regard to $x[y]$.

3 Second Order ODEs : Further nonlinear effects.

A generic second order differential equation is

$$F[x, y[x], y'[x], y''[x]] == 0$$

and save for a couple of special tricks the general idea is to reduce it to a first-order equation using an appropriate **infinitesimal symmetry** of the equation, to solve the reducent equation and then to "pull back" the solutions.

"Pulling back" the solutions of the first-order reducent equation is always possible up to quadratures. This is a task with the complexity equal to the complexity of solving another first-order ordinary differential equation when an integrating multiplier for that equation is known.

Therefore technically the worst obstacles for "pulling back" solutions are:

- non-elementary integrations, leading to quadratures in the final answer,
- transcendental equations, leading to suspended `Solve`.

We have seen examples of both already in the previous section.

There is more to be said however about the level of the reduced equation.

Here we encounter the following possibilities:

- The reduced equation is not solvable
- It is solvable but the solution is implicit; there are obstacles to solving the equation
- It is solvable and the solution is explicit

Examples of all the possible combinations of these situations are given below.

■ Example 5. (Kamke 6.14)

```
In[14]:= y''[x] == Exp[y[x]]

                    y[x]
Out[14]= y''[x] == E

In[15]:= DSolve[%,y[x],x]

                                y[x]
DBG::DInputstatus {{y''[x] == E    }, {}, {y[x]},

  {}, {2}}
DBG::puriflag False
Yes, DBG  is here
DBG::DInputStatus {{Calculus`DSolve`Private`var1$74

      '[Calculus`DSolve`Private`z$74] ==

     Calculus`DSolve`Private`z$74
    E                            /

     Calculus`DSolve`Private`var1$74[Calculus`DSolv\

       e`Private`z$74]}, {},

  {Calculus`DSolve`Private`var1$74[Calculus`DSolve`\

     Private`z$74]}, {}, {1}}
DBG::->LSFIQ<- memo {{Calculus`DSolve`Private`var1$\

      74[Calculus`DSolve`Private`z$74] ->

            Calculus`DSolve`Private`z$74
    Sqrt[2 E                            + C[1]]},

  {Calculus`DSolve`Private`var1$74[Calculus`DSolve`\

      Private`z$74] ->

             Calculus`DSolve`Private`z$74
    -Sqrt[2 E                            + C[1]]}}

Solve::ifun:
   Warning: Inverse functions are being used by
     Solve, so some solutions may not be found.

Solve::ifun:
   Warning: Inverse functions are being used by
     Solve, so some solutions may not be found.

Out[15]= {{y[x] ->

                        Sqrt[C[1]] (-x - C[2]) 2
        C[1] (-1 + Tanh[---------------------] )
                                  2
    Log[----------------------------------------]}\
                          2

   , {y[x] ->

                        Sqrt[C[1]] (x + C[2]) 2
        C[1] (-1 + Tanh[--------------------] )
                                  2
    Log[---------------------------------------]}}
                         2
```

Everything is OK on all levels here.

■ Example 6. (Kamke 6.17)

```
In[16]:= y''[x] == a Sin[y[x]]

Out[16]= y''[x] == a Sin[y[x]]
```

```
In[17]:= DSolve[%,y[x],x]

DBG::DInputstatus {{y''[x] == a Sin[y[x]]}, {},
  {y[x]}, {}, {2}}
DBG::puriflag False
Yes, DBG  is here
DBG::DInputStatus {{Calculus`DSolve`Private`var1$94
      '[Calculus`DSolve`Private`z$94] ==
    (a Sin[Calculus`DSolve`Private`z$94]) /
     Calculus`DSolve`Private`var1$94[Calculus`DSolv\
       e`Private`z$94]}, {},
  {Calculus`DSolve`Private`var1$94[Calculus`DSolve`\
     Private`z$94]}, {}, {1}}
DBG::->LSFIQ<- memo {{Calculus`DSolve`Private`var1$\
      94[Calculus`DSolve`Private`z$94] ->
    Sqrt[C[1] - 2 a Cos[Calculus`DSolve`Private`z$\
         94]]}, {Calculus`DSolve`Private`var1$94
      [Calculus`DSolve`Private`z$94] ->
    -Sqrt[C[1] - 2 a
        Cos[Calculus`DSolve`Private`z$94]]}}
                                             1
Out[17]= {-Integrate[-------------------------,
                     Sqrt[C[1] - 2 a Cos[y[x]]]
     y[x]] == x + C[2],
                             1
  Integrate[-------------------------, y[x]] ==
            Sqrt[C[1] - 2 a Cos[y[x]]]
   x + C[2]}
```

The reduced first-order equation is solved here but non-elementary quadratures are introduced while "pulling back" the solutions. Out[108] is however a well-defined general solution of the original equation though expressed in an implicit form.

■ Example 7 (Kamke 6.93)

```
In[18]:= y''[x] == a ( x y'[x] - y[x] )^2 / x^3

                                     2
                 a (-y[x] + x y'[x])
Out[18]= y''[x] == -------------------
                            3
                           x

In[19]:= DSolve[%,y[x],x]

DBG::DInputstatus {{y''[x] ==
                         2
    a (-y[x] + x y'[x])
    -------------------}, {}, {y[x]}, {}, {2}}
             3
            x
DBG::puriflag False

Solve::ifun:
   Warning: Inverse functions are being used by
     Solve, so some solutions may not be found.

Solve::ifun:
   Warning: Inverse functions are being used by
     Solve, so some solutions may not be found.

                                  (a y[x])/x
Out[19]= Solve[Log[x (1 + a E            C[2])] -
    a y[x]
    ------ == C[1], y[x]]
      x
```

Everything solved fine until the very end, where a transcendental equation has been encountered.

■ Example 8 (Kamke 6.50)

```
In[20]:= y''[x] == - b y[x] - a y[x] y'[x]

Out[20]= y''[x] == -(b y[x]) - a y[x] y'[x]

In[21]:= DSolve[%,y[x],x]

DBG::DInputstatus {{y''[x] ==

    -(b y[x]) - a y[x] y'[x]}, {}, {y[x]}, {}, {2}}
DBG::puriflag False
Yes, DBG  is here
DBG::DInputStatus {{Calculus`DSolve`Private`var1$137

      '[Calculus`DSolve`Private`z$137] ==

    (-(b Calculus`DSolve`Private`z$137) -

       a Calculus`DSolve`Private`z$137

        Calculus`DSolve`Private`var1$137[Calculus`D\

          Solve`Private`z$137]) /

     Calculus`DSolve`Private`var1$137[Calculus`DSol\

       ve`Private`z$137]}, {},

  {Calculus`DSolve`Private`var1$137[Calculus`DSolve\

     `Private`z$137]}, {}, {1}}
                                      2
DBG::->LSFIQ<- memo Solve[-(a

                                  2
       Calculus`DSolve`Private`z$137 ) +

    2 b Log[b + a Calculus`DSolve`Private`var1$137

          [Calculus`DSolve`Private`z$137]] -

    2 a Calculus`DSolve`Private`var1$137[Calculus`D\

       Solve`Private`z$137] == C[1],

  Calculus`DSolve`Private`var1$137[Calculus`DSolve`\

    Private`z$137]]
And now DBG is here ...

                                           2     2
Out[21]= {Solve[2 b Log[b + a y'[x]] - a  y[x]  -

     2 a y'[x] == C[1], y'[x]]}
```

A transcendental equation for the $y'[x]$ has been encountered on a low level of the reduced equation. This is an obstacle for getting actual solutions. It is output explicitly by the `DSolve`.

The useful content of this output is that the above family of the first order equations is equivalent to the original second-order equation.

■ Example 9 (Kamke 6.169)

This is one of the equations that has been difficult to solve with the original version of the Calculus/DSolve package (as included in version 2.1 of *Mathematica*) but that is cleanly solved by the advanced version.

```
In[22]:= y''[x] == ( y[x] y'[x] - x y'[x]^2 ) / ( x y[x] )

                                         2
                    y[x] y'[x] - x y'[x]
Out[22]= y''[x] == ---------------------
                           x y[x]
```

```
In[23]:= DSolve[%,y[x],x]

DBG::DInputstatus {{y''[x] ==

                          2
    y[x] y'[x] - x y'[x]
    ---------------------}, {}, {y[x]}, {}, {2}}
          x y[x]
DBG::puriflag False
DBG::DInputStatus {{Calculus`DSolve`Private`u$156

           Calculus`DSolve`Private`u$156[x]
  '[x] == -------------------------------- -
                          x

                                     2
    2 Calculus`DSolve`Private`u$156[x] }, {},

   {Calculus`DSolve`Private`u$156[x]}, {}, {1}}
DBG::DInputStatus {{Calculus`DSolve`Private`u$156

           Calculus`DSolve`Private`u$156[x]
  '[x] == -------------------------------- -
                          x

                                     2
    2 Calculus`DSolve`Private`u$156[x] }, {},

   {Calculus`DSolve`Private`u$156[x]}, {}, {1}}

                                C[1]  2
Out[23]= {{y[x] -> Sqrt[-1 + E      x ] C[2]}}
```

4 Conclusion

Thus, because the nonlinear case is it itself more difficult, the output of DSolve is not so plain as for the linear case. Sometimes more cooperation is needed on the part of the user to finalize the answer. Sometimes the answers does not appear at all because of general insolvability of the equation or because of the limitations of the solving techniques or because of technical limitations in some cases.

However the evaluation results briefly listed in the appendix show that definite progress has been made in the field of the nonlinear equations since *Mathematica*2.0 version.

5 Appendix: Evaluating the New DSolve

The evaluation is based on the equations from the E. Kamke reference book [1].

Among the first order nonlinear equations referred in Chapter I of the reference book there are exactly 12 that have closed form solutions but are not solved by the current DSolve. These are the equations with reference numbers: 1.21, 1.22 , 1.32 , 1.46 , 1.54 , 1.62 , 1.80 , 1.207 , 1.212 , 1.236 , 1.238 , 1.249 .

The general problem with these equations is that they require special tricks for obtaining the solution.

The behavior of the present DSolve is not so exhaustive in the case of nonlinear second order ODE Equations with closed form solutions are rather easily found in the sixth chapter of Kamke's book that are not solved by DSolve as yet.

DSolve is better however than dsolve in the current version of Maple V.

Here are just few examples of equations the dsolve fails to solve while DSolve is able to find the general solution:

Kamke 6.63

$$y''[x] == a\,(y'[x]^2 + 1)^{3/2}$$

Kamke 6.79

$$y''[x] == x\,y'[x]^2 + 2\,y'[x]/x + y[x]^2/x == 0$$

Kamke 6.93

$$y''[x] == a\,(x\,y'[x] - y[x])^2/x^3$$

Kamke 6.97

$$y''[x] == (2\,x\,y[x] + x^3)\,y'[x]/x^4 - 4\,y[x]^2/x^4$$

Kamke 6.98

$$y''[x] == (x^2\,y'[x]^2 - x^3\,y'[x] - 4\,y[x]^2)/x^4$$

Kamke 6.99

$$x^4\,y''[x] + (x\,y'[x] - y[x])^3 == 0$$

Kamke 6.122

$$y''[x] == (y'[x]^2 - f[x]\,y[x]\,y'[x] - g[x]\,y[x]^2)/y[x]$$

Macsyma's ODE2 also seems to fail on all the above examples.

(Maxima 4.153 of August 19, 1990 has been tested.)

References

[1] E. Kamke, Differentialgleichungen, Lösungsmethoden und Lösungen, Band 1, Gewöhnliche Differentialgleichungen, Chelsea Publishing Company, New York, 1959.