

Design and Implementation of ODESolve 1+ : An Enhanced REDUCE ODE Solver

Francis J. Wright
School of Mathematical Sciences
Queen Mary and Westfield College
University of London

<http://www.maths.qmw.ac.uk/~fjw/>

Abstract and transparencies on the web in PDF

29 April 1999

Latest release in <http://reduce.maths.qmw.ac.uk/packages/odesolv1>
which contains full source, tests and documentation.

Current *release* is version 1.06, but I will describe version 1.065.

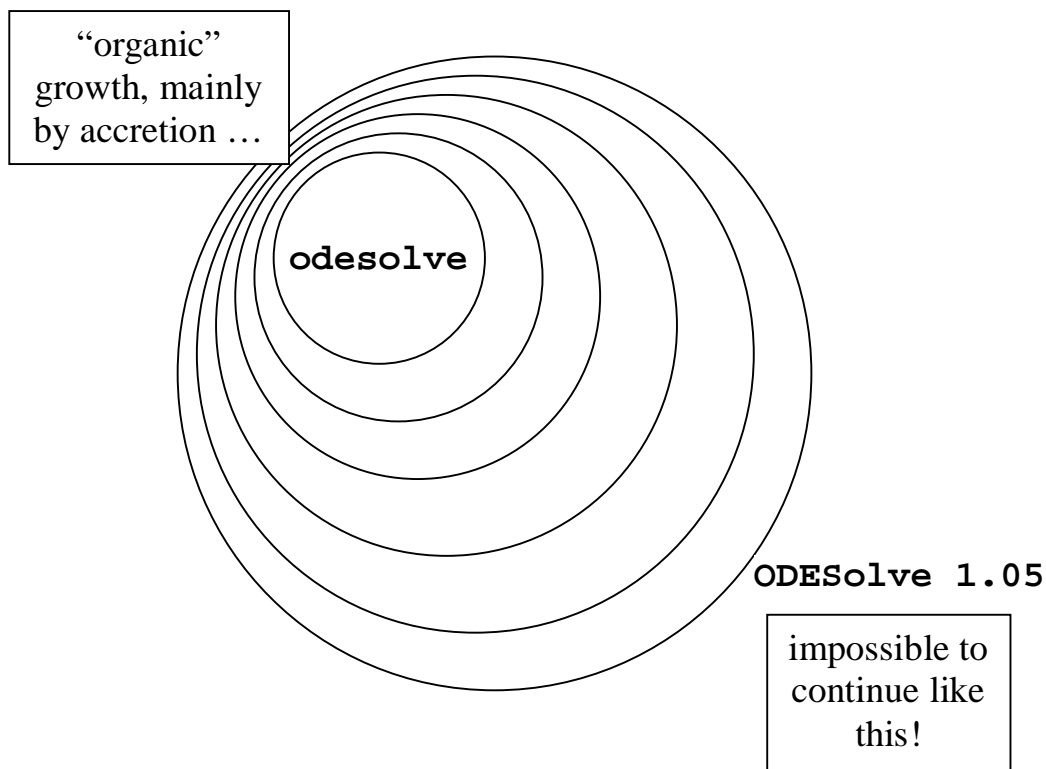
Requires REDUCE development version or *next* release version.

Outline of Talk

- History
- ODESolve 1.065
- ODE Classification
- Linear / Nonlinear
- ODESolve 1.065 Logic (Simplified)
- Nonlinear first-order solution techniques
- Nonlinear higher-order solution techniques
- Interface
- Details of some nonlinear first-order solution techniques
- Solution of second-order linear special function ODEs

History

Developed from Malcolm MacCallum's 1990 `odesolve` by fixes and additions ...



ODESolve 1.06 is completely restructured, mainly to split linear and nonlinear ODEs and classify ODEs

- *firstly* by linearity, and
- *secondly* by order.

Most procedures are recursively callable.

ODESolve 1.065

Module	Lines	Function
odesolv1	150	Header, tracing, etc.
odeintfc	650	User interface and condition code
odetop	550	Top level ODESolve routines
odelin	650	Simple linear ODEs
odespcfn	600	Linear special function ODEs
odenon1	750	Special form nonlinear ODEs of order 1
odenonn	400	Special form nonlinear ODEs of order > 1
odepatch	150	Temporary REDUCE patches and extensions
Total	4000	(cf. 800 lines for original <code>odesolve</code>)

ODE Classification

Not unique, e.g.

$$\frac{dy}{dx} = y$$

is both *linear first-order* and *first-order separable*!

ODESolve 1.06+

First order linear	First order nonlinear
Higher order linear	Higher order nonlinear

Linear first order	Nonlinear first order
Linear higher order	Nonlinear higher order

Linear / Nonlinear

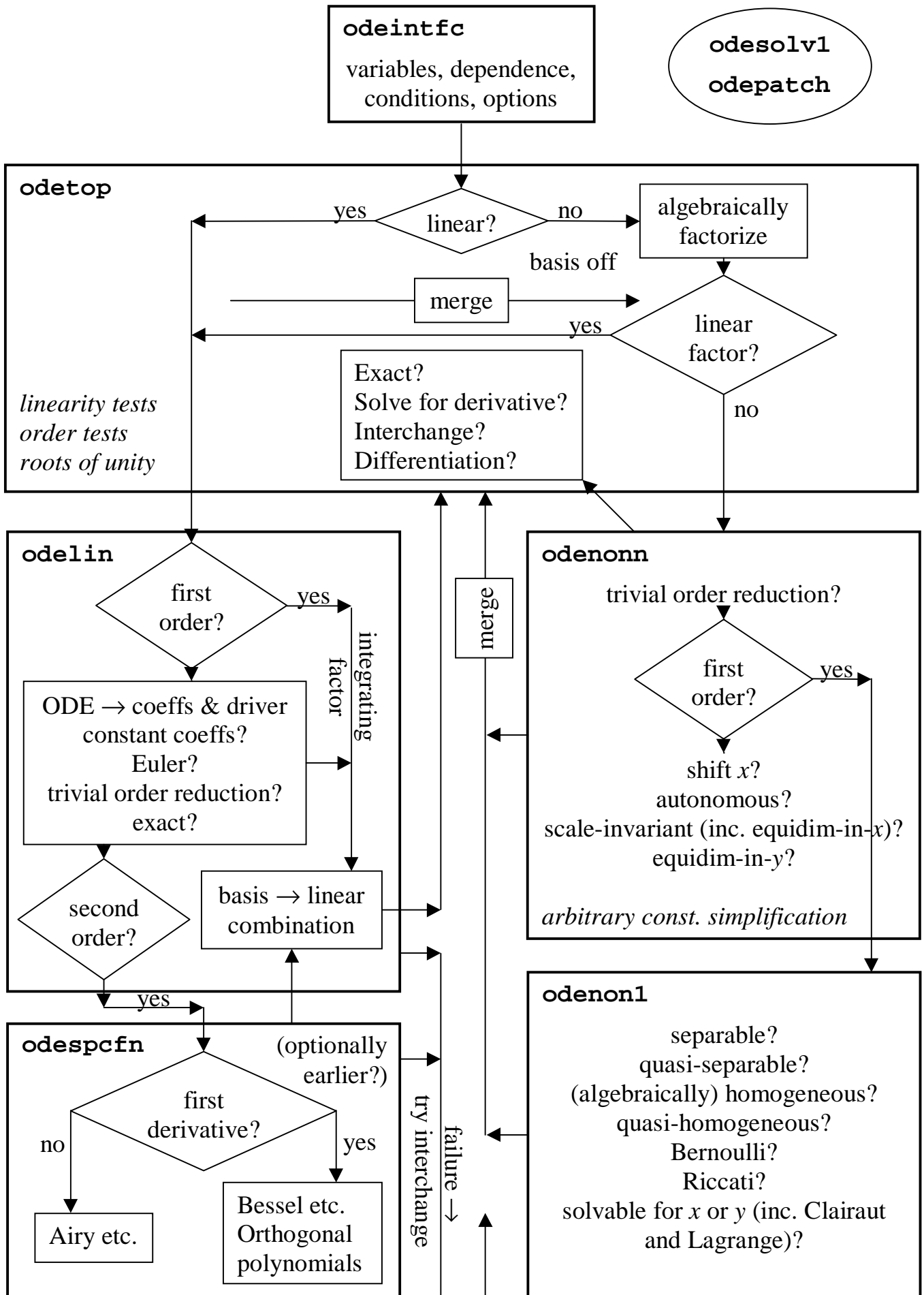
Why merge? (ODESolve \leq 1.05)

- ◆ Common preprocessing:-
 - algebraic factorization
 - solving for single derivative
 - trivial order reduction (y missing, etc)
 - shifting x
- ◆ Common solution techniques, e.g. “equidimensional-in- x ” ~ Euler.

Why distinguish? (ODESolve \geq 1.06)

- ◆ Represent linear ODE solutions as *basis* and *particular integral*
 - output option and internal representation
- ◆ Represent linear ODE using *coefficients* and *driver*
- ◆ Optimized preprocessing for linear ODEs
 - factorization \rightarrow “monic” coefficients
 - trivial order reduction can use multiple integration to raise order because solution is guaranteed explicit
 - shift x only if necessary, e.g. Euler and special function ODEs; much more reliable.

ODESolve 1.065 Logic (Simplified)



Nonlinear first-order solution techniques

First degree

1. **Separable:** $\frac{dy}{dx} = f(x)g(y) \rightarrow$ quadrature
2. **Quasi-separable:** $\frac{dy}{dx} = f(y+kx)$ cf. “quasi-homogeneous”
3. **Homogeneous:** $\frac{dy}{dx} = f(y/x) \rightarrow$ first-order *linear* ODE
4. **Quasi-homogeneous:** $\frac{dy}{dx} = f\left(\frac{a_1x+b_1y+c_1}{a_2x+b_2y+c_2}\right)$, f arbitrary!
5. **Bernoulli:** $\frac{dy}{dx} = P(x)y + Q(x)y^n$, n arbitrary \rightarrow first-order *linear* ODE
6. **Riccati:** $\frac{dy}{dx} = a(x)y^2 + b(x)y + c(x) \rightarrow$ second-order *linear* ODE

Higher degree

- ◆ “Solvable for y ” $\rightarrow y = f(x, y') \rightarrow$ ODE for $y'(x)$
Special cases:
 - Lagrange: $y = xF(y') + G(y') \rightarrow$ first-order *linear* ODE for $x(y')$
 - Clairaut: $F(xy' - y) = G(y')$, $y' \rightarrow$ arbitrary constant gives general solution, differentiating may give singular solution
- ◆ “Solvable for x ” $\rightarrow x = f(y, y') \rightarrow$ ODE for $y'(y)$

Nonlinear higher-order solution techniques

“Simplifiers”:

- ◆ explicit absence of y and low order derivatives \rightarrow trivial order reduction
- ◆ manifest dependence on $x+k \rightarrow$ shift

Special cases of Lie symmetry.

Classification is not unique, so try the most useful first!

1. **Autonomous:** no explicit dependence on $x \rightarrow$ ODE in y' of order one lower.
2. **Scale invariant** or **Equidimensional in x :** invariant under $x \rightarrow ax$, $y \rightarrow a^p y$.
Exponential transformation of $x \rightarrow$ autonomous \rightarrow ODE of order one lower.
(Equidimensional in x if $p = 0$, cf. Euler linear ODE.)
3. **Equidimensional in y :** invariant under $y \rightarrow ay$. Exponential transformation of $y \rightarrow$ ODE of the same order that *may* be “more linear”.
(All reduced linear ODEs are trivially equidimensional in y .)

Interface

REDUCE Development Version, Thu Feb 18 11:54:05 1999 ...

```
1: load_package odesolv1;
```

```
ODESolve 1.065 loading ...
```

Variables and Dependence

```
2: ODESolve(df(y,x) = y);
```

```
*** Dependent var(s) assumed to be y
```

```
*** Independent var assumed to be x
```

```
*** depend y , x
```

```
      x  
{y=e *arbconst(1)}
```

```
3: operator y;
```

```
4: ODESolve(df(y(x),x) = y(x));
```

```
*** Dependent var(s) assumed to be y(x)
```

```
*** Independent var assumed to be x
```

```
      x  
{y(x)=e *arbconst(2)}
```

```
5: ODESolve(df(y,x) = p, y, x);
```

```
{y=arbconst(3) + p*x}
```

```
6: depend {p,q},u, {u},v, {v},x;
```

```
7: ODESolve(df(y,x) = p, y, x);
```

```
{y=arbconst(4) + int(p,x)}
```

```
8: operator p;
```

```
9: ODESolve(df(y,x) = p(x), y, x);
```

```
{y=arbconst(5) + int(p(x),x)}
```

10: ODESolve(df(y,x) + p*y = q*y^n, y, x);

$$\left\{ \frac{y^n}{y} = \left(e^{\int(p,x) \cdot n} \cdot \text{arbconst}(6) - e^{\int(p,x) \cdot n} \cdot \frac{\int \left(\frac{e^{\int(p,x)} \cdot q}{e^{\int(p,x) \cdot n}}, x \right) \cdot n}{e^{\int(p,x) \cdot n}} \right)^{\frac{1}{n-1}} \right.$$

$$\left. + e^{\int(p,x) \cdot n} \cdot \frac{\int \left(\frac{e^{\int(p,x)} \cdot q}{e^{\int(p,x) \cdot n}}, x \right) / e^{\int(p,x) \cdot n}}{e^{\int(p,x) \cdot n}} \right\}$$

11: ODESolve(df(y,x) + p*y = q*y^3, y, x);

$$\left\{ \frac{1}{y^2} = e^{2 \cdot \int(p,x)} \cdot \text{arbconst}(7) - 2 \cdot e^{2 \cdot \int(p,x)} \cdot \frac{\int \left(\frac{q}{e^{2 \cdot \int(p,x)}}, x \right)}{e^{2 \cdot \int(p,x)}} \right\}$$

Options

explicit, expand, full, implicit, noint, verbose, basis, tracing

12: ODESolve(df(y,x) + p*y = q*y^3, explicit);

*** Dependent var(s) assumed to be y
 *** Independent var assumed to be x

$$\left\{ y = \frac{\text{plus_or_minus}(\text{tag}_1)}{e^{\int(p,x)} \cdot \sqrt{\text{arbconst}(9) - 2 \cdot \int \left(\frac{q}{e^{2 \cdot \int(p,x)}}, x \right) / e^{2 \cdot \int(p,x)}}} \right\}$$

13: ODESolve(df(y,x) + p*y = q*y^3, y, x, full, tracing);

This is a nonlinear ODE of order 1.

It is of Bernoulli type.

$$\left\{ y = \frac{1}{e^{\int(p,x)} \cdot \sqrt{\text{arbconst}(11) - 2 \cdot \int \left(\frac{q}{e^{2 \cdot \int(p,x)}}, x \right) / e^{2 \cdot \int(p,x)}}} \right.,$$

$$\left. y = \frac{-1}{e^{\int(p,x)} \cdot \sqrt{\text{arbconst}(11) - 2 \cdot \int \left(\frac{q}{e^{2 \cdot \int(p,x)}}, x \right) / e^{2 \cdot \int(p,x)}}} \right\}$$

Conditions

```
14: ODESolve(df(y,x) = y, y = a, x = 0);
```

$$\{y=e^x * a\}$$

```
15: ODESolve(df(y,x) = y, y = a, x = 0, tracing);
```

This is a linear ODE of order 1.

It is solved by the integrating factor method.

General solution is $\{y=e^x * \text{arbconst}(13)\}$

Applying conditions $\{\{x=0, y=a\}\}$

$$\{y=e^x * a\}$$

```
16: ODESolve(df(y,x,2) + y, y, x, {{y=0, x=0}, {y=a, x=pi/2}});
```

$$\{y=\sin(x)*a\}$$

```
17: ODESolve(df(y,x,2) + y, y, x, {y=0, df(y,x)=a, x=0}, tracing);
```

This is a linear ODE of order 2.

It has constant coefficients.

General solution is $\{y=\text{arbconst}(17)*\sin(x) + \text{arbconst}(16)*\cos(x)\}$

Applying conditions $\{\{x=0, y=0, df(y,x)=a\}\}$

$$\{y=\sin(x)*a\}$$

Not robust, needs better support for functional notation and eigenvalue problems!

Details of some nonlinear first-order solution techniques

Separable: $\frac{dy}{dx} = F(x, y) = f(x)g(y) \rightarrow$ quadrature

Any separation of $F(x, y)$ will suffice! Choose any constant α such that $F(\alpha, y)$ exists and $F(\alpha, y) \neq 0$. If $F(x, y) = f(x)g(y)$ then $F(\alpha, y) = f(\alpha)g(y)$ and

$\frac{F(x, y)}{F(\alpha, y)} = \frac{f(x)}{f(\alpha)}$ is free of y . (*Implicit dependence requires care!*)

Quasi-separable: $\frac{dy}{dx} = F(x, y) = f(y + kx)$ cf. “quasi-homogeneous”

$F(x, y) = f(y + kx)$ iff $\frac{\partial F/\partial x}{\partial F/\partial y} = k = \text{constant}$.

Quasi-homogeneous: $\frac{dy}{dx} = F(x, y) = f\left(\frac{a_1x + b_1y + c_1}{a_2x + b_2y + c_2}\right)$, f arbitrary!

1. Find the first “function argument” that is a rational function, with numerator and denominator that both depend on x .
2. Check that numerator and denominator have the same degree in x (and y).
If that degree > 1 then extract the squarefree factors using $p/\text{gcd}(p, p')$.
3. Check that numerator and denominator are really *linear polynomials* in x and y (ignoring dependence of y on x).
The ODE was quasi-homogeneous iff the new ODE is homogeneous.
(The degenerate case $a_1b_2 = a_2b_1$ was already treated as quasi-separable.)

Solution of second-order linear special function ODEs

Part of the top-level special function solver:

```
c0 := first odecoeffs1;
soln := if (c1 := second odecoeffs1) then <<
  traceodel "First-order derivative present.";
  symbolic or(
    % y" - 2x*y' + 2n*y:
    ODESolve!-Hermite(c1, c0, x),
    % x*y" + (m+1-x)*y' + n*y:
    % x*y" + (beta-x)*y' - alpha*y:
    ODESolve!-Kummer!-Laguerre(c1, c0, x),
    % x^2*y" + x*y' + (+/-x^2-n^2)*y:
    % x^2*y" + (1-2alpha)x*y' +
    % (+/-beta^2 gamma^2 x^(2gamma) + (alpha^2-n^2 gamma^2))*y:
    ODESolve!-QuasiBessel(c1, c0, x, driver1),
    % (x^2-1)*y" + (alpha-beta+(alpha+beta+2)*x)*y' -
    % n*(n+alpha+beta+1)*y:
    ODESolve!-Jacobi(c1, c0, x),
    % x*(x-1)*y" + ((alpha+beta+1)*x-gamma)*y' -
    % alpha*beta*y:
    ODESolve!-Gauss(c1, c0, x))
  >> else <<
  traceodel "First-order derivative missing.";
  symbolic if depends(den c0, x) then
    % x^2*y" + (+/-beta^2 gamma^2 x^(2gamma) + (1/4-n^2 gamma^2))*y
    ODESolve!-QuasiRiccatiBessel(c0, x) or
    % y" + (-1/4+k/x+(1/4-m^2)/x^2)*y
    ODESolve!-Whittaker(c0, x)
  else
    % y" + x*y
    ODESolve!-Airy(c0, x) or
    % y" + ((2n+1)-x^2)*y
    %% ODESolve!-Weber!-Hermite(c0, x) or
    % y" - (1/4x^2+A)*y
    ODESolve!-Parabolic!-Cylinder(c0, x)
  >>;
```

A very simple explicit solution routine:

```
algebraic procedure ODESolve!-Airy(c0, x);
%% y" + x*y
%% c0 = a^2(ax+b)
begin scalar coeffs, a;
  traceodel "Airy equation?";
  coeffs := coeff(c0, x); % {a^2b, a^3}
  if high_pow neq 1 or depends(coeffs, x) then return;
  a := (second coeffs)^(1/3);
  x := -(a*x + (first coeffs)/a^2);
  traceode "Airy equation.";
  return {Airy_Ai(x), Airy_Bi(x)}
end$
```