

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
E-mail: F.J.Wright@QMW.ac.uk
<http://www.maths.qmw.ac.uk/~fjw/>

21 April 1999

CATHODE Workshop, Marseilles, May 1999

`ODESolve 1+` is a project to enhance the `odesolve` package that was written primarily by Malcolm MacCallum about 10 years ago and is currently distributed with `REDUCE`. I anticipate that the new ODE solver will be mature enough to distribute with releases of `REDUCE after 3.7` (the next release). In the meantime, the latest release of `ODESolve 1+` is always available from <http://reduce.maths.qmw.ac.uk/packages/odesolv1>, which includes full source code, test data and documentation. It currently runs only with the `REDUCE` development version, although it will run with `REDUCE 3.7` when it is released. It uses a (non-standard) technique that should allow the package to be built easily using *exactly* the same file set in either the development or a released version of `REDUCE`.

`REDUCE` allows a variable y to depend explicitly on a variable x through the operator syntax $y(x)$, or implicitly by simply declaring y to depend on x . Implicit dependence is much closer to the way one normally works on paper, and is the form used internally. However, any other symbolic parameter appearing in an ODE may also depend either explicitly or implicitly on any other variable, and this dependence can in principle be nested arbitrarily deeply. Handling such dependence correctly is an interesting challenge, and adds considerably to the complexity of some of the code. (This issue does not arise in Maple, for instance, because Maple has no implicit dependence mechanism.) Although the `ODESolve 1+` interface supports systems of ODEs there is still no code to solve them.

`ODESolve 1+` versions from 1.00 to 1.05 were essentially organic developments around `odesolve`, whereas the latest release version 1.06 is a substantial re-implementation that distinguishes *firstly* between linear and nonlinear ODEs and *secondly* between first and higher order. Most of `ODESolve 1+` can be (and is) called recursively (unlike `odesolve`).

The version of `ODESolve 1+` that I will describe is about half way between the current release version 1.06 and the next release version 1.07. It consists of about 4000 lines of code (including comments) distributed among 8 files as follows. The code is split roughly equally between linear and nonlinear ODEs.

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 ODE solvers
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

The solution techniques currently implemented are entirely “classical” (i.e. they are mostly described in Zwillinger’s “Handbook of Differential Equations”).

The (new) top-level module “odetop” checks linearity. A nonlinear ODE is first algebraically factorized, and the factors rechecked for linearity. Linear ODEs are instead made “monic” by dividing through by the coefficient of the highest derivative, and are represented internally by their coefficient functions and the “driver” term if there is one. Solutions of linear ODEs are represented internally in terms of a basis for their solution space together with a particular integral if there is one.

The linear and first-order nonlinear solution techniques implemented are standard “mathematical techniques”, although I try to avoid potentially expensive high-level operations such as factorization. The higher-order nonlinear techniques are special cases of Lie symmetry. If all else fails, I interchange the dependent and independent variables. (I do this even for linear ODEs, which makes them nonlinear. I would prefer not to, and it breaks the basis representation, but it leads to solutions that are not accessible to my current set of linear techniques!)

All components of the linear solver allow affine transformations of the independent variable, which are handled separately by each solution technique (rather than as a general simplification). `ODESolve 1+` now uses a uniform decision procedure, in which the classification and solution code is merged and each solution technique returns either a solution or nil. This avoids a lot of repetition and allows a much cleaner structure. However, it may be harder to handle partial solutions, an issue that I have not yet fully resolved.

I currently use whichever of algebraic or symbolic mode is more convenient. Most of the code is in algebraic mode, except for the interface module and a number of small utility routines. For example, the routines to test linearity and compute order are written in symbolic mode and operate on REDUCE’s internal representations. At some future date, some of the algebraic mode code will be converted to symbolic mode for improved efficiency.

I am currently working on the “odespcfn” module, which recognises a collection of standard second-order linear “special function” ODEs. This includes most of those listed in the “Handbook of Mathematical Functions” (edited by Abramowitz and Stegun), such as Bessel’s equation, various hypergeometric equations and the equations satisfied by the classical orthogonal polynomials. Although this is clearly a pattern matching problem, I finally abandoned lengthy attempts to use both of REDUCE’s general pattern matchers in favour of special purpose code.