Solving Stiff ODEs

Home

Background

Non-stiff Solvers

Overview
Implementations

Stiff Solvers

Rosenbrock
BSBD semi-implicit
Implicity RK
BDF methods

LSODE and VODE

LSODE and ODEPACK
VODE and CVODE
LSODA vs. CVODE

Comparisons

Conclusion

Download lsoda.c

Background

Stiff ODEs are evil. Most of numerical methods for solving ordinary differential equations will become unbearably slow when the ODEs are stiff. Unfortunately, a large set of ODEs are frequently stiff in practice. It is very important to use an ODE solver that solves stiff equations efficiently.

Solvers for non-stiff equations

An overview of methods

Most of the classical numerical methods for solving ODEs are for non-stiff equations only. These methods include the simplest Euler method, the most widely used fourth-order Runge-Kutta method (RK4), the adaptive Runge-Kutta-Fehlberg method (RKF), the adaptive Runge-Kutta-Cash-Karp method (RKCK), the adaptive Runge-Kutta-Dormand-Prince method and the multi-step Adams' method (predictor-corrector method). You can find the detailed description of RK4 and RKCK methods in Numerical Recipes or other similar textbooks. Euler and Runge-Kutta methods are one-step methods in that they refer only to one previous value to determine the current value. Adams' method, which refers to several previous function values, is quite different and is probably less known to people, but this method may potentially incur few function calls [reference].

Implmentations

It is fairly easy to implement the 4th order Runge-Kutta method. Adaptive methods are not hard, either. You can largely read the textbook and write your own version. The GNU Scientific Library (GSL) has implemented all the one-step methods above. The interface is also very convenient.

Solvers for stiff equations

Solvers for stiff equations usually require to evaluate the Jacobian, which can be very expensive. Sophisticated solvers usually automatically switch between stiff and non-stiff methods to achieve good performance in both cases. Please read the next section for the well known implementations.

Rosenbrock Methods

This method is described in Numerical Recipe. It is relatively simple to implement, but its performance is quickly degraded when we require higher precision (beyond ~1e-5), which is confirmed by my experience. This is not a good general-purpose

method.

Apparently, for stiff equations, XPP uses the Rosenbrock method.

Bulirsch-Stoer-Bader-Deuflhard semi-implicit methods

Numerical Recipe (2nd edition) describes this method as "an excellent routine for all stiff problems", although "very occasionally" it "will encounter a singular matrix". Unfortunately, given my ODEs, this case frequently occurs to the implementation in the book.

GSL comes with an alternative implementation (namely bsimp) of this method. This implementation does not complain about singular matrix, but incurs much more function calls than RKCK on one of my examples. Apparently, this is not a good general-purpose method, either.

Implicit Runge-Kutta methods

GSL also provides the implicit 2nd/4th order Runge-Kutta methods. These methods, however, do not seem to outperform the explicit methods (see below).

Backward Differentiation Formulae (BDF or Gear methods)

Different from the above methods, BDF is a multi-step method. It is probably the most widely used method for stiff equations. Numerical Algorithms with C gives a good implementation apparently.

LSODE and VODE

LSODE and VODE are probably the best known ODE solvers, both in Fortran and developed by the Lawrence Livermore National Laboratory. LSODE eventually evolves into ODEPACK, while VODE into CVODE as part of the Sundials package. Older versions of ODEPACK and CVODE can also be found at the netlib repository.

LSODE and ODEPACK

According to this page, ODEPACK consists of LSODE and its eight variants. LSODA is one of them, which automatically switches between the Adams' method and the BDF method. A C version of LSODA is available here, but few people are using that to my knowledge. I further combined the source codes, fixed a few minor bugs and added a more convenient (but less versatile) interface. You can download my version here. The complete documentation for the Fortran version are available at netlib, from odepack or from CCL.NET. According to the tiny benchmark below, LSODA greatly outperforms other simple stiff or non-stiff methods.

By the way, Mathematica also combine the Adams method and the BDF method to solve ODEs [reference].

VODE and CVODE

VODE was developed after LSODE. Its C version, CVODE which is included in the Sundials package, is believed to be the best ODE solver for C programs. An older version of CVODE can be found here. I do not know how much it differs from the latest version. Apparently, CVODE does not automatically choose between stiff and

non-stiff methods [reference].

LSODA vs. CVODE

VODE was developed later and in its paper, the author claimed that VODE outperformed LSODE. That paper was published over 20 years ago and a lot have happened to the two packages. I do not know how the two programs are compared now and am not aware of any systematic benchmark. An implicit comparison is actually done by the SOSlib developers. In a presentation, they showed SOSlib, which uses CVODE, achieves similar performance to Copasi which uses LSODA.

A (very rough) comparison on ONE set of ODEs

I have a set of ODEs describing 41 genes/proteins and 63 reactions between them. The system approaches to a fixed point at t=100.0 approximately. I run several ODE solvers on these ODEs and checked how many iterations and function evaluations are required to integrate to t=100.0. The following table shows the results. Methods with a star require a user-provided function to calculate the Jacobian. I did not count the function calls on calculating the Jacobian.

Method	#iterations	#evaluations
nr-rkck	1,272	8,772
nr-stifbs*	failed	failed
nr-stiff*	18,272	54,960
lsoda	381	1,754
gsl-rkck	1,275	11,664
gsl-rkf45	1,315	11,852
gsl-rk2	2,559	13,677
gsl-rk4	961	15,034
gsl-rk8pd	902	16,884
gsl-rk2imp	2,765	56,262
gsl-rk4imp	1,129	33,038
gsl-gear1	>9,999	110,060
gsl-gear2	1,267	36,784
gsl-bsimp*	458	79,914

Lsoda is clearly the best on these ODEs. It requires far fewer function evaluations than the rest. Nr-rkck outperforms gsl-rkck/rkf45 probably due to the details in implementation. RKCK and RKF45 are also good candidates given their simplicity. Gsl-rk8pd uses the 8-9th order RKPD method. It approaches the fixed point with fewer iterations than RKCK, but requires more function evaluations and thus is less efficient. I am not sure if gsl-gear1/2 and gsl-rk2/4imp are suitable for stiff equations. Apparently they do not seem to be. Gsl-bsimp and nr-stiff are both

designed for stiff ODEs and both require user-provided routines to calcuate the Jacobian. However, they are not comparable to the others on this example.

Please note that the performance of a solver highly depends on the characteristics, in particular the stiffness, of the input ODEs. You will probably come to a different conclusion given another set of ODEs.

Conclusion

LSODA is a good ODE solver especially when you do not know whether your ODEs are stiff or not. One potential concern comes from its implementation. The C version of LSODA have many global variables, which makes it unsuitable for solving more than one set of ODEs. A better implementation should pack all the global variables into a struct.

My version of LSODA can be downloaded here.

Last modified: 2009-05-22