Software for the numerical integration of ODE by means of high-order Taylor methods (III)

Àngel Jorba
angel@maia.ub.es

University of Barcelona

Advanced Course on Long Term Integrations

Outline

```
Usage: ./taylor
 [-name ODE_NAME]
 [-o outfile]
 [-doubledouble | -qd_real | -dd_real | -gmp -gmp_precision PRECISION]
 [-main | -header | -jet | -main_only]
 [-step STEP_CONTROL_METHOD]
 [-u | -userdefined] STEP_SIZE_FUNCTION_NAME ORDER_FUNCTION_NAME
 [-f77]
 [-sqrt]
 [-headername HEADER_FILE_NAME]
 [-debug] [-help] [-v]  file
```

Main C call:

```
int taylor_step_ODE_NAME(MY_FLOAT *time,
                         MY_FLOAT *xvars,
                         int      direction,
                         int      step_ctrl_method,
                         double   log10abserr,
                         double   log10relerr,
                         MY_FLOAT *endtime,
                         MY_FLOAT *stepused,
                         int      *order)
```

Main Fortran 77 call:

```
void taylor_f77_ODE_NAME__(MY_FLOAT *time,
                           MY_FLOAT *xvars,
                           int      *direction,
                           int      *step_ctrl_method,
                           double   *log10abserr,
                           double   *log10relerr,
                           MY_FLOAT *endtime,
                           MY_FLOAT *stepused,
                           int      *order,
                           int      *flag)
```

Next, we will perform long term integrations on the spatial RTBP, but now using gmp.

We will integrate the same initial condition as before, for 10^6 units of time.

The goal is to check the propagation of roundoff errors when the basic arithmetic is changed.

```
value of H at the initial condition: -0.1336207158e1
numerical integration starts...
1  0.183827140545086 174 -0.8204174804320 2e-153
2  0.374344795509428 174 -0.5966672584960 1e-153
3  0.574965855180706 174 -0.6712506658080 1e-153
4  0.789299521404807 174 -0.1491668146240 0e-153
5  1.000000000000000 174 -0.2237502219360 0e-153
iterates: 5  final time: 1.000000e+00
```

Numerical integration using gmp with a 512 bits mantissa and asking for a relative error of 10^{-150}.

```
value of H at the initial condition: -0.1336207158e1
numerical integration starts...
1  0.18007100738854 4 0.124236998889749e-303
2  0.36649219119811 4 0.678258138919457e-303
3  0.56247621401637 4 0.878726168201663e-303
4  0.77152718140379 4 0.921848099579533e-303
5  0.99716668197227 4 0.106043126217200e-302
6  1.00000000000000 4 0.106043682485665e-302
iterates: 6  final time: 1.000000e+00
```

As before, but using a 1024 bits mantissa and asking for a relative error of 10^{-300}.

We want to compare our implementation of Taylor method against a few well known methods.

A characteristic of these methods is that they have a freely available implementation, which is the one we have used. These implementations are coded in FORTRAN77, which adds an extra difficulty on the comparisons, since the observed differences may come from the different compilers.

Therefore, to help the readers with these comparisons, the package includes the code for all the examples, so that they can be run on any combination of compiler/computer for comparisons.

These tests have been done in a GNU/Linux workstation, with an Intel Pentium III processor running at 500 MHz. We have used the GNU compilers gcc and g77, version 2.95.4.

The methods considered are

- dop853, an explicit Runge-Kutta code of order 8,
- odex, an extrapolation method of varying order based on the Gragg-Bulirsh-Stoer algorithm.

Both methods are documented in the book by Hairer, Nørsett and Wanner (2000), and the code we have used can be downloaded from

`http://www.unige.ch/math/folks/hairer/software.html`

We note that extrapolation methods are similar to Taylor in the sense that they can use arbitrarily high orders, so they are the natural methods to compare with.

For the tests, we have used three vector fields: the RTBP, the Lorenz system, a periodically forced pendulum, and the RTBP. The equations for the Lorenz system are

$$\begin{aligned} \dot{x} &= 10(y - x), \\ \dot{y} &= x(28 - z) - y, \\ \dot{z} &= xy - \frac{8}{3}z, \end{aligned}$$

and the equations for the forced pendulum are

$$\begin{aligned} \dot{x} &= y, \\ \dot{y} &= -\sin(x) - 0.1y + 0.1\sin(t) \end{aligned}$$

Given an initial condition, we compute the corresponding orbit during, say, 16 units of time and to compare the final point with the true value to obtain the real absolute error.

The true value has been obtained from an integration with the Taylor method using the gmp arithmetic with mantissas of 128 and 256 bits.

In the next tables we show the computer time and final error for the three methods, using different thresholds for the step size control. To have a measurable running time, the program repeats the same calculation 1000 times.

Lorenz								
dop583			odex			taylor		
ε	time	error	ε	time	error	ε	time	error
1.e-10	7.01	5.9e-03	1.e-10	8.73	6.2e-02	1.e-10	7.61	3.1e-06
1.e-11	8.91	5.0e-04	1.e-11	10.11	3.3e-03	1.e-11	7.99	4.4e-07
1.e-12	11.65	4.3e-05	1.e-12	11.54	2.0e-04	1.e-12	8.40	4.8e-08
1.e-13	15.31	3.7e-06	1.e-13	12.74	5.8e-06	1.e-13	8.80	3.3e-08
1.e-14	20.19	1.2e-06	1.e-14	15.04	6.4e-06	1.e-14	9.22	3.4e-08
1.e-15	26.76	8.9e-07	1.e-15	17.81	3.7e-06	1.e-15	9.75	9.2e-09
1.e-16	35.51	9.5e-07	1.e-16	50.47	1.9e-06	1.e-16	10.75	7.5e-09

Perturbed pendulum								
dop583			odex			taylor		
ε	time	error	ε	time	error	ε	time	error
1.e-10	0.62	3.4e-11	1.e-10	1.49	6.9e-10	1.e-10	0.38	2.8e-13
1.e-11	0.78	3.6e-12	1.e-11	1.70	4.9e-11	1.e-11	0.42	2.1e-14
1.e-12	1.03	3.1e-13	1.e-12	1.93	1.7e-12	1.e-12	0.44	7.6e-15
1.e-13	1.38	2.7e-14	1.e-13	2.17	9.1e-14	1.e-13	0.47	1.2e-15
1.e-14	1.83	2.3e-15	1.e-14	2.36	4.4e-15	1.e-14	0.48	8.7e-16
1.e-15	2.45	2.1e-15	1.e-15	2.68	3.1e-15	1.e-15	0.52	5.8e-16
1.e-16	3.24	3.2e-15	1.e-16	3.09	1.1e-14	1.e-16	0.59	3.8e-16

RTBP								
dop583			odex			taylor		
ε	time	error	ε	time	error	ε	time	error
1.e-10	1.43	1.1e-09	1.e-10	1.74	1.8e-09	1.e-10	1.68	6.2e-12
1.e-11	1.84	9.4e-11	1.e-11	2.02	9.2e-11	1.e-11	1.86	4.6e-13
1.e-12	2.44	8.6e-12	1.e-12	2.43	2.4e-11	1.e-12	2.08	4.4e-14
1.e-13	3.24	8.0e-13	1.e-13	2.74	3.7e-13	1.e-13	2.27	7.2e-15
1.e-14	4.32	7.5e-14	1.e-14	3.14	1.5e-13	1.e-14	2.50	4.2e-15
1.e-15	5.73	9.9e-15	1.e-15	3.71	2.4e-13	1.e-15	2.82	1.7e-15
1.e-16	7.63	2.0e-15	1.e-16	4.85	1.3e-13	1.e-16	3.26	5.8e-15

ADOL-C is a public domain package for automatic differentiation. The main differences between the automatic differentiation of our package and ADOL-C are:

- ADOL-C is a general purpose package, while `taylor` is specifically designed for the numerical integration of ODEs.
- The input of ADOL-C is a C/C++ function (with some restrictions in the grammar used), while `taylor` has its own input grammar, which is a bit more restrictive.
- ADOL-C does not include code for the step size control. This means that ADOL-C can only be used to generate the Taylor coefficients and the user must supply code for the order and step size control. For this reason, we will only test the speed of the generation of the Taylor coefficients.

As before, the tests have been done on an Intel Pentium III running at 500 MHz, using ADOL-C version 1.8.7. The examples considered are the Lorenz system, RTBP, the Lorenz system and a periodically forced pendulum. To measure the time, we have computed the jet of derivatives 100,000 times.

As before, the tests have been done on an Intel Pentium III running at 500 MHz, using ADOL-C version 1.8.7. The examples considered are the Lorenz system, RTBP, the Lorenz system and a periodically forced pendulum. To measure the time, we have computed the jet of derivatives 100,000 times.

	degree	Lorenz	Pendulum	RTBP
ADOL-C	40	92.82	140.57	403.22
Taylor	40	3.59	3.43	14.75
ADOL-C	20	24.44	34.82	87.99
Taylor	20	1.13	1.07	4.65
ADOL-C	10	9.13	11.58	26.20
Taylor	10	0.41	0.39	1.62

Here we will illustrate one of the uses of extended arithmetic: the computation of small quantities defined as the difference of very close numbers.

Here we will illustrate one of the uses of extended arithmetic: the computation of small quantities defined as the difference of very close numbers.

Let us consider the dynamical system

$$\ddot{x} - \sin(x) = \mu \sin\left(\frac{t}{\varepsilon}\right),$$

where μ and ε are small parameters. When $\mu = 0$, $x = 0$ and $x = 2\pi$ are hyperbolic points such that the stable and unstable manifolds of $x = 0$ coincide with the unstable and stable manifolds of $x = 2\pi$. For $\mu > 0$ and small, the points $x = 0$ and $x = 2\pi$ become hyperbolic periodic orbits and their invariant manifolds do not coincide but intersect transversally

It is usual to take the section $t = 0 \mod 2\pi\varepsilon$ so that the ODE becomes a conservative 2-D map, with hyperbolic fixed points near $x = 0$ and $x = 2\pi$.

Due to the symmetries of the problem, the unstable manifold of $x = 0$ intersects transversally the stable manifold of $x = 2\pi$ at $x = \pi$.

Here we will compute the intersection angle of these manifolds for $\mu = \varepsilon = 0.04$. The methods used here will be quite simple, since the only goal is to illustrate the capabilities provided by `taylor`.

First, we use `taylor` to produce a time stepper for the ODE.

It is not difficult to write the 2-D map defined by the stroboscopic section $t = 0 \mod 2\pi\varepsilon$. The differential of this map is given by the numerical integration of the variational flow, again by means of the Taylor method.

We have asked `taylor` to call the `dq` library for the arithmetic, using the `qd_real` type (it provides nearly 64 decimal digits).

Next, we code a Newton method to obtain the two hyperbolic fixed points near $x = 0$ and $x = 2\pi$, and the eigenvalues and eigenvectors of the differential of the map at these points (in fact, due to the symmetries of the problem, it is enough to perform these computations for one of them).

The next step is to use the eigenvalues as a (linear) approximation to the manifolds and to grow them till they cut the line $x = \pi$. At this point we have used two different procedures:

- To obtain a table of values of the two manifolds on a mesh of points x_j around $x = \pi$, and to use numerical differentiation (with 3 steps of extrapolation) to approximate the intersection angle between the two manifolds.
- To compute an initial point p, at an approximated distance of 10^{-25} from the fixed point, that it is mapped on the line $x = \pi$ after a certain number of iterates. Then we have used the corresponding eigenvector at the fixed point as the tangent vector to the manifold at the initial point p . Then we have iterated this point and the vector to obtain an approximation to the tangent vector of the manifolds at $x = \pi$.

The agreement between the two approaches allows to conclude that the intersection angle is $2.769781155284039017022 \times 10^{-17}$ (we only write the common digits).

We note that the computation is not difficult provided one has an efficient procedure to integrate ODEs in extended precision.