Taylor series based computations and MATLAB ODE solvers comparisons

Václav Šátek, Jiří Kunovský, , Filip Kocina, and , and Jan Chaloupka

Citation: AIP Conference Proceedings **1558**, 2289 (2013); doi: 10.1063/1.4825997
View online: http://dx.doi.org/10.1063/1.4825997
View Table of Contents: http://aip.scitation.org/toc/apc/1558/1
Published by the American Institute of Physics

Taylor Series Based Computations and MATLAB ODE Solvers Comparisons

Václav Šátek*, Jiří Kunovský†, Filip Kocina† and Jan Chaloupka†

*IT4Innovations, VŠB Technical University of Ostrava,
17. listopadu 15/2172, 708 33 Ostrava-Poruba, Czech Republic [1]

†University of Technology, Faculty of Information Technology,
Božetěchova 2, 612 66 Brno, Czech Republic [2]

Abstract. The paper is a part of student cooperation in AKTION project (Austria-Czech). Taylor series method for solving differential equations represents a non-traditional way of a numerical solution. Even though this method is not much preferred in the literature, experimental calculations done at the Department of Intelligent Systems of the Faculty of Information Technology of TU Brno have verified that the accuracy and stability of the Taylor series method exceeds the currently used algorithms for numerically solving differential equations.

The paper deals with possibilities of numerical solution of Initial Value Problems of Ordinary Differential Equations (ODEs) - using the Taylor series method with automatic computation of higher Taylor series terms.

The explicit and implicit scheme of Taylor series method is compared with numerical solvers implemented in MATLAB software [1]. The computation time and accuracy of our approach are compared with that of MATLAB ode solvers on a set of ODEs test examples [2].

Keywords: Differential equations, Taylor series method, Stiff systems, TKSL, MATLAB
PACS: 02,89

INTRODUCTION

The "Modern Taylor Series Method" (MTSM) is used for numerical solution of differential equations. The MTSM is based on a recurrent calculation of the Taylor series terms for each time interval. Thus the complicated calculation of higher order derivatives (much criticised in the literature) need not be performed but rather the value of each Taylor series term is numerically calculated. Solving the convolution operations is another typical algorithm used.

An important part of the MTSM is an automatic integration order setting, i.e. using as many Taylor series terms as the defined accuracy requires. Thus it is usual that the computation uses different numbers of Taylor series terms for different steps of constant length.

The MTSM has been implemented in TKSL software [3]. Some articles that are focused on the MTSM were published last years [4, 5].

There are several papers that focus on computer implementations of the Taylor series method in different context "a variable order and variable step" (see, for instance, [6, 7]). Another more detailed description of a variable step size version and software implementation of the Taylor series method can be seen in [8]. The stability domain for several Taylor methods is presented in [9]. Promising A-stable combination of implicit Taylor series method with Trapezoidal rule is described in [10].

[1] vaclav.satek@vsb.cz
[2] kunovsky@fit.vutbr.cz

11th International Conference of Numerical Analysis and Applied Mathematics 2013
AIP Conf. Proc. 1558, 2289-2292 (2013); doi: 10.1063/1.4825997

EXPLICIT SOLVERS

The best-known and most accurate method of calculating a new value of a numerical solution of ordinary differential equation $y' = f(t,y),\ \ y(0) = y_0$ is to construct the Taylor series [11, 12].

The $n-$th order method ($ORD = n$) uses n Taylor series terms in the explicit form

$$y_{i+1} \quad = \quad y_i + h \cdot f(t_i, y_i) + \frac{h^2}{2!} \cdot f^{[1]}(t_i, y_i) + \cdots + \frac{h^n}{n!} \cdot f^{[n-1]}(t_i, y_i)\,, \quad ORD = n\,. \tag{1}$$

$$\tag{2}$$

Non-Stiff Problems

Benchmark set of ordinary differential equations [2] was used for tests. Following comparisons of entire "non-stiff package", except for non-continuous equations F1 - F5 have been completed. Simulated interval (t_{max}) was set to 100 seconds, minimal integration time step (h) was set to 1 second.

MTSM (based on explicit Taylor series form (2)) was compared with MATLAB explicit **ode** solvers. It was found that the best MATLAB solver for our set of test examples was **ode45** solver. Results of computation times comparisons for MTMS (**expTay**) $ORD = 10$ can be seen in Tab. 1 (every time is taken from 50 runs of computations). Ratios of computation times

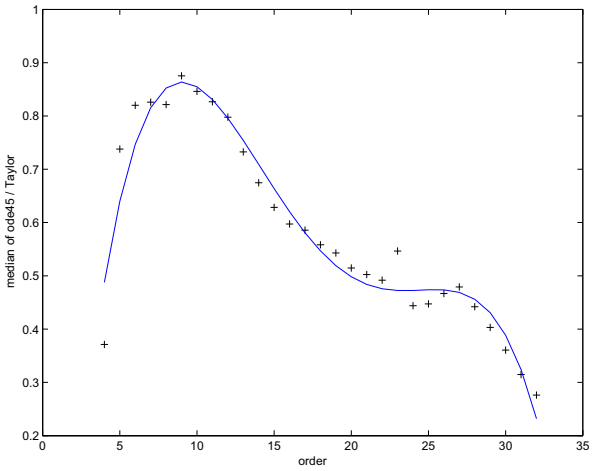
$$ratio_{1e} = \frac{ode45}{expTay}, \qquad ratio_{2e} = \frac{expTay}{ode45}$$

can be seen in Tab. 1 (the bold numbers represents faster solutions of computations).

Median time of all examples for different order of MTSM can be seen in figure right to Tab. 1.

TABLE 1. Comparison of **ode45** and explicit **Taylor** ($ORD = 10$)

	ode45 [s]	**expTay** [s]	$ratio_{1e}$	$ratio_{2e}$
A1	0.0118	0.0053	**2.2108**	0.4523
A2	0.0047	0.0055	0.8541	**1.1708**
A3	0.0304	0.0164	**1.8603**	0.5375
A4	0.0048	0.0050	0.9571	**1.0448**
A5	1.8295	0.0349	**52.3957**	0.0191
B1	0.0485	0.0315	**1.5404**	0.6492
B2	0.0222	0.0086	**2.5713**	0.3889
B3	0.0104	0.0064	**1.6132**	0.6199
B4	0.0365	1.0375	0.0351	**28.4578**
B5	0.0240	0.0032	**7.4430**	0.1344
C1	0.0165	0.0123	**1.3407**	0.7459
C2	0.0710	0.4187	0.1696	**5.8964**
C3	0.0378	0.0216	**1.7494**	0.5716
C4	0.0552	0.0918	0.6012	**1.6633**
D1	0.0553	0.7054	0.0784	**12.7506**
D2	0.0603	0.6933	0.0870	**11.4892**
D3	0.0716	0.7111	0.1007	**9.9329**
D4	0.0932	0.7062	0.1320	**7.5786**
D5	0.1269	1.9798	0.0641	**15.6022**
E1	0.0259	0.0552	0.4693	**2.1308**
E2	0.0581	0.0564	**1.0302**	0.9707
E3	0.0271	0.1009	0.2682	**3.7290**
E4	0.0042	0.0050	0.8381	**1.1932**
E5	0.0129	0.0338	0.3811	**2.6243**
median			0.9595	3.2340



IMPLICIT SOLVERS

There are some peculiar systems of differential equations, which cannot be solved by commonly used (explicit) numerical methods - *the stiff systems*. While the definition of this kind of systems is intuitively clear to the mathematicians the exact definition has not been yet specified [13].

Similarly like explicit Taylor series the implicit form of the $n-$th order method ($ORD = n$) can be expresed in the form

$$y_{i+1} = y_i + h \cdot f(t_{i+1}, y_{i+1}) - \frac{h^2}{2!} \cdot f'(t_{i+1}, y_{i+1}) - \cdots - \frac{(-h)^n}{n!} \cdot f^{(n-1)}(t_{i+1}, y_{i+1}), \;\; ORD = n. \tag{3}$$

Stiff Problems

Implicit Taylor Series Method with Recurrent Calculation of Taylor Series Terms and Newton Method (ITMRN) based on (3) was implemented in MATLAB. Stability and accuracy of the ITMRN computation of ODEs were analyzed.

Benchmark set of ordinary differential equations [2] was used for tests. Comparisons of entire "stiff package" equations A1, A3, A4, B1-B5 where analytic solution is well-known (obtained from Maple software [14]) have been completed. Simulated interval was used from article, integration time step was set to time interval (just 1 integration step was needed).

TABLE 2. Comparison of MATLAB stiff ode solvers and ITMRN

	ode15s [s]	**impTay1** [s]	**impTay2** [s]	**impTay3** [s]	$ratio_{1i}$	$ratio_{2i}$	$ratio_{3i}$	**‖Error(y)‖** (*ORD*)
A1	0.0538	0.0022	0.0023	0.0015	34.1588	58.8119	36.5149	3.2468×10^{-5} (10)
A3	0.0722	0.0023	0.0024	0.0015	45.9286	83.017842	50.9011	1.064661×10^{-5} (10)
A4	0.0920	0.0129	0.0069	0.0037	24.6389	67.185681	28.1318	3.063018×10^{-5} (6)
B1	0.5533	0.0012	0.0010	0.00063	872.6757	12678.4996	7307.7485	0.7003512×10^{-5} (3)
B2	0.0608	0.0043	0.0044	0.0026	22.9871	59.6822	28.7128	3.248772×10^{-5} (10)
B3	0.0651	0.0043	0.0045	0.0027	23.5398	63.9215	30.2985	3.248772×10^{-5} (10)
B4	0.1377	0.0044	0.0046	0.0027	50.4536	228.7384	105.1956	3.248772×10^{-5} (10)
B5	1.4354	0.0044	0.0045	0.0027	525.8835	1388.9469	594.2142	3.248772×10^{-5} (10)

Three types of ITMRN were implemented:

- Implicit Taylor series method + Newton's method where Jacobian matrix was computed using **differential formulae** - **impTay1** in Tab. 2
- Implicit Taylor series method + Newton's method where Jacobian matrix was computed with **symbolic operations** (analytically) - **impTay2** in Tab. 2
- Implicit Taylor series method + Newton's method where Jacobian matrix was computed using **Broyden's method** - **impTay3** in Tab. 2

Absolute error $||\text{Error}(y)||$ in Tab. 2 of numerical solution is defined as difference between numerical y_i and analytical $y(t_i)$ solution

$$||\text{Error}(y)|| = ||y_i - y(t_i)||, \tag{4}$$

where $t_i = h \cdot i$ (for ITMRN and our test examples in Tab. 2 - $i = 1$ and $h = t_{max}$). Maximum absolute error in our tests is set to 10^{-4}. Absolute error is completed with information of (*ORD*) used.

Time of computations of **ode15**, **impTay1**, **impTay2**, **impTay3** are presented for test equations in Tab. 2. Ratios of computation times for others MATLAB implicit solvers

$$ratio_{1i} = \frac{ode15s}{impTay3}, \qquad ratio_{2i} = \frac{ode25s}{impTay3}, \qquad ratio_{3i} = \frac{ode23tb}{impTay3},$$

can be also seen in Tab. 2.

Condition number of Jacobian matrix in Newton iteration method becomes higher, that is why multiple arithmetic is needed for higher order and higher integration step size. [3]

CONCLUSIONS

Ten Taylor series terms ($ORD = 10$) of explicit Taylor series proved to be the best method for the given set of "non-stiff" ODEs. As expected, a solution of linear systems of ODEs using higher order Taylor series (and corresponding higher integration time step) was faster than that of low order Taylor series method (lower integration step).

Multiple arithmetic for higher order of implicit Taylor series method (and higher integration step) must be used as condition number of Jacobian matrix in Newton iteration method becomes higher.

Detailed information will be given during the ICNAAM 2013 conference.

ACKNOWLEDGMENTS

This paper has been elaborated in the framework of the project New creative teams in priorities of scientific research, reg. no. CZ.1.07/2.3.00/30.0055 (CZ.1.05/1.1.00/02.0070), supported by Operational Programme Education for Competitiveness and co-financed by the European Social Fund and the state budget of the Czech Republic. The paper includes the solution results of the Ministry of Education, Youth and Sport research project No. MSM 0021630528 and the international AKTION research project Number 64p13.

REFERENCES

1. T. MathWorks, *MATLAB and Simulink for Technical Computing* (2013), URL `http://www.mathworks.com[online]`.
2. W. H. Enright, and J. D. Pryce, "Two FORTRAN packages for assessing initial value methods," in *ACM Trans. Math. Softw.*, ACM, 1987, vol. 13, pp. 1–27, ISSN 0098-3500.
3. J. Kunovský, *Modern Taylor Series Method*, FEI-VUT Brno, 1994, habilitation work.
4. J. Kunovský, M. Pindryč, V. Šátek, and V. Zbořil, F., "Stiff systems in theory and practice," in *Proceedings of the 6th EUROSIM Congress on Modelling and Simulation*, ARGE Simulation News, 2007, p. 6, ISBN 978-3-901608-32-2.
5. J. Kunovský, V. Šátek, V. Vopěnka, and A. Schirrer, "Stiffness in Technical Initial Problems," in *Proceedings of the 10th International Conference of Numerical Analysis and Applied Mathematics*, 1479, American Institute of Physics, 2012, p. 4, ISBN 978-0-7354-1091-6, ISSN 1551-7616.
6. R. Barrio, F. Blesa, and M. Lara, "High-precision numerical solution of ODE with high-order Taylor methods in parallel," in *Monografías de la Real Academia de Ciencias de Zaragoza*, 2003, vol. 22, pp. 67–74.
7. R. Barrio, F. Blesa, and M. Lara, "VSVO Formulation of the Taylor Method for the Numerical Solution of ODEs," in *Computers and Mathematics with Applications*, 2005, vol. 50, pp. 93–111.
8. A. Jorba, and M. Zou, "A software package for the numerical integration of ODE by means of high-order Taylor methods," in *Exp. Math.*, 2005, vol. 14, pp. 99–117.
9. R. Barrio, "Performance of the Taylor series method for ODEs/DAEs," in *Applied Mathematics and Computation*, 2005, vol. 163, pp. 525–545, iSSN 00963003.
10. X. Chang, H. Zheng, and X. Gu, "An A-stable improved Taylor series method for power system dynamic-stability simulation," in *Proceedings of the Power and Energy Engineering Conference Asia-Pacific*, 2010, pp. 1–5, ISBN 978-1-4244-4812-8.
11. E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I*, vol. Nonstiff Problems. Springer-Verlag Berlin Heidelberg, 1987, ISBN 3-540-56670-8.
12. E. Hairer, and G. Wanner, *Analysis by Its History*, corrected third printing. Springer-Verlag New York Berlin Heidelberg, 2000, ISBN 0-387-94551-2.
13. E. Hairer, and G. Wanner, *Solving Ordinary Differential Equations II*, second revised ed. with 137 Figures, vol. Stiff and Differential-Algebraic Problems. Springer-Verlag Berlin Heidelberg, 2002, ISBN 3-540-60452-9.
14. Maplesoft, *Technical Computing Software for Engineers, Mathematicians, Scientists, Instructors and Students* (2013), URL `http://www.maplesoft.com[online]`.
15. GMP, *The GNU MP Bignum Library* (2013), URL `http://gmplib.org/[online]`.

[3] The special data type `vpa` (variable precision arithmetic) in MATLAB is implemented or for C/C++ programs GMP library [15] should be used (this library is used in TKSL software).