Available online at www.sciencedirect.com

ScienceDirect

IFAC-PapersOnLine 48-1 (2015) 693–694

# Taylor Series Based Solution of Linear ODE Systems and MATLAB Solvers Comparison

V. Šátek *,** F. Kocina ** J. Kunovský ** A. Schirrer ***

* IT4Innovations, VŠB Technical University of Ostrava, 17. listopadu 15/2172, 708 33 Ostrava-Poruba, Czech Republic, (e-mail: vaclav.satek@vsb.cz)
** Brno University of Technology, Faculty of Information Technology, Božetěchova 2, 612 66 Brno, Czech Republic
*** Institute of Mechanics and Mechatronics, Vienna University of Technology, Getreidemarkt 9, 1060 Vienna, Austria

**Abstract:** The Modern Taylor Series Method (MTSM) is employed here to solve initial value problems of linear ordinary differential equations. An automatic computation of higher Taylor series terms and an efficient, vectorized coding of explicit and implicit schemes enables a very fast computation of the solution to specified accuracy. For a set of benchmark problems from literature, the MTSM significantly outperforms standard solvers. Finally, ideas of parallelizing the MTSM computations are discussed.



*Keywords:* Ordinary differential equations, Initial value problems, Taylor series, MATLAB

## 1. 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. 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 the TKSL software package (Kunovský, 1994).

Several papers focus on computer implementations of the Taylor series method in a variable-order and variable-step context (see, for instance, Barrio et al. (2005), the TIDES software implemented in Wolfram (2014), or in Jorba and Zou (2005)). The reduction of rounding errors (Rodríguez and Barrio, 2012) and utilization of multiple arithmetics (Barrio et al., 2011) improve the applicability of Taylor series based algorithms.

This paper demonstrates that the MTSM, specialized to directly solving linear ODE systems, solves non-stiff and stiff systems very fast (in explicit and implicit formulations, respectively) and outperforms standard solvers in the considered benchmark problems.

## 2. EXPLICIT SCHEME OF TAYLOR SERIES

In this article, we have focused on effective solution of linear systems of ODEs using Taylor series scheme. 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 (Hairer et al., 1987).

The $n-$th order method uses $n$ Taylor series terms in the explicit form

$$\begin{aligned} y_{i+1} &= y_i + hf(t_i, y_i) + \frac{h^2}{2!} f^{[1]}(t_i, y_i) + \cdots \\ &+ \frac{h^n}{n!} f^{[n-1]}(t_i, y_i). \end{aligned} \tag{1}$$

Equation (1) for linear systems of ODEs in the form $\boldsymbol{y}' = \boldsymbol{A}\boldsymbol{y} + \boldsymbol{b}$ could be rewritten

$$\begin{aligned} \boldsymbol{y}_{i+1} &= \boldsymbol{y}_i + h(\boldsymbol{A} \cdot \boldsymbol{y}_i + \boldsymbol{b}) + \frac{h^2}{2!} \boldsymbol{A}(\boldsymbol{A}\boldsymbol{y}_i + \boldsymbol{b}) + \\ &\cdots + \frac{h^n}{n!} \boldsymbol{A}^{(n-1)}(\boldsymbol{A}\boldsymbol{y}_i + \boldsymbol{b}), \end{aligned} \tag{2}$$

where $\boldsymbol{A}$ is the constant Jacobian matrix and $\boldsymbol{b}$ is the constant right-hand side.

Vectorized MATLAB code of explicit Taylor series **expTay** with a variable order and variable step size scheme for linear systems of ODEs (2) has been implemented. This algorithm was compared on a set of "non-stiff" linear systems (see Enright and Pryce (1987)) with vectorized MATLAB explicit **odeNN** solvers. Benchmarking results are shown in Table 1 (each reported runtime is the median value of 100 computations). Ratios of computation times $ratio_e = \boldsymbol{ode23/expTay} > 1$ indicate faster computation of the MTSM in all test cases. Exact solutions were obtained by the Maple software package (Maplesoft, 2014). All solvers' tolerances were set to obtain relative and absolute tolerances of $10^{-4}$ with respect to the exact solutions.

## 3. IMPLICIT SCHEME OF TAYLOR SERIES

The implicit Taylor series scheme for linear systems of ODEs are constructed as follows:


Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2015.05.210

Table 1. Median computation time: explicit Taylor **expTay** and MATLAB explicit **odeNN** solver comparison

| | **ode23** [s] | **ode45** [s] | **ode113** [s] | **expTay** [s] | $ratio_e$ |
|---|---|---|---|---|---|
| A1 | 0.00497 | 0.00537 | 0.00751 | 0.000831 | 5.98 |
| B2 | 0.00633 | 0.00758 | 0.0128 | 0.00218 | 2.9 |
| C1 | 0.00653 | 0.00574 | 0.0111 | 0.00114 | 5.72 |
| C2 | 0.01 | 0.0147 | 0.0277 | 0.00651 | 1.54 |
| C3 | 0.00636 | 0.00805 | 0.0156 | 0.003 | 2.11 |
| C4 | 0.00679 | 0.00836 | 0.0166 | 0.00359 | 1.89 |

$$\begin{aligned} \boldsymbol{y}_{i+1} &= \boldsymbol{y}_i + h(\boldsymbol{A}\boldsymbol{y}_{i+1} + \boldsymbol{b}) - \frac{h^2}{2!}\boldsymbol{A}(\boldsymbol{A}\boldsymbol{y}_{i+1} + \boldsymbol{b}) - \\ &\cdots - \frac{(-h)^n}{n!}\boldsymbol{A}^{(n-1)}(\boldsymbol{A}\boldsymbol{y}_{i+1} + \boldsymbol{b}) . \end{aligned} \tag{3}$$

Implicit Taylor series method with recurrent calculation of Taylor series terms and Newton method (**impTay**) based on (3) was implemented in MATLAB using vectorization. The Jacobian matrix is computed using Broyden's method.

A benchmark problem set of "stiff" linear ODEs from Enright and Pryce (1987) was used for tests. Comparisons of the problems A1, A3, A4, and B1-B5 whose analytic solutions are known (from the Maple software package (Maplesoft, 2014)) have been completed. The simulated intervals were adopted from Enright and Pryce (1987), and the integration time step was set to the entire time interval (just 1 integration step was needed). Relative and absolute tolerances for the computations were again set to $10^{-4}$. Comparisons of MATLAB "stiff" **odeNNs** solvers with **impTay** are shown in Table 2. High ratios of computation times $ratio_i = \boldsymbol{ode15s/impTay}$ show that the MTSM method significantly outperforms the standard solvers.

Table 2. Time of solutions: implicit Taylor **impTay** and MATLAB implicit **odeNNs** solvers comparisons

| | **ode15s** [s] | **ode23s** [s] | **ode23tb** [s] | **impTay** [s] | $ratio_i$ |
|---|---|---|---|---|---|
| A1 | 0.0605 | 0.169 | 0.101 | 0.0003 | 194.6 |
| A3 | 0.085 | 0.243 | 0.144 | 0.00001 | 263 |
| A4 | 0.111 | 0.478 | 0.192 | 0.0003 | 294.8 |
| B1 | 0.268 | 1.473 | 0.8 | 0.0003 | 244.4 |
| B2 | 0.069 | 0.285 | 0.134 | 0.00003 | 172.4 |
| B3 | 0.073 | 0.308 | 0.146 | 0.00003 | 211.4 |
| B4 | 0.117 | 0.549 | 0.242 | 0.00003 | 348.4 |
| B5 | 1.155 | 1.529 | 0.664 | 0.00003 | 3306.7 |

## 4. PARALLEL IMPLEMENTATION

As can be seen from (2), each term of Taylor Series for a linear system can be computed independently. So their computation can be distributed into multiple computation units (utilizing a distributed memory architecture). Hence thread $j \in \{1 \dots m\}$ evaluates

$$\boldsymbol{A}_j = \sum_{k=0}^{\frac{n}{m}-1} \frac{h^{mk+j}}{(mk+j)!}\boldsymbol{A}^{mk+j-1} \tag{4}$$

and the final sum is computed afterwards. Therefore expression (2) can be transformed to

$$\boldsymbol{y}_{i+1} = \left(\left(\sum_{j=1}^{m}\boldsymbol{A}_j\right)\boldsymbol{A} + \mathbf{I}\right)\boldsymbol{y}_i + \left(\sum_{j=1}^{m}\boldsymbol{A}_j\right)\boldsymbol{b} \tag{5}$$

where $\mathbf{I}$ is the identity matrix.

## 5. CONCLUSION

The Taylor series scheme is highly efficient in solving linear ODEs. It significantly outperforms standard solvers on the considered benchmark problems. Results for double precision arithmetics and a maximum Taylor series order of 90 have been shown. Multiple arithmetics is needed for higher orders. Future studies will address the efficiency and scalability of MTSM ODE solvers in different parallelization architectures.

## ACKNOWLEDGEMENTS

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 (as well as the the IT4Innovations Centre of Excellence 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 international AKTION research project Number 69p22 and the internal BUT projects FIT-S-12-1 and FIT-S-14-2486.

## REFERENCES

Barrio, R., Blesa, F., and Lara, M. (2005). VSVO Formulation of the Taylor Method for the Numerical Solution of ODEs. In *Computers and Mathematics with Applications*, volume 50, 93–111.

Barrio, R., Rodríguez, M., Abad, A., and Blesa, F. (2011). Breaking the limits: The taylor series method. *Applied Mathematics and Computation*, 217, 7940–7954.

Enright, W.H. and Pryce, J.D. (1987). Two fortran packages for assessing initial value methods. In *ACM Trans. Math. Softw.*, volume 13, 1–27. ACM.

Hairer, E., Nørsett, S.P., and Wanner, G. (1987). *Solving Ordinary Differential Equations I.* vol. Nonstiff Problems. Springer-Verlag Berlin Heidelberg. ISBN 3-540-56670-8.

Jorba, A. and Zou, M. (2005). A software package for the numerical integration of ODE by means of high-order Taylor methods. In *Exp. Math.*, volume 14, 99–117.

Kunovský, J. (1994). *Modern Taylor Series Method.* FEI-VUT Brno. Habilitation work.

Maplesoft (2014). *MAPLE software.* URL `http://www.maplesoft.com [online]`.

MathWorks, T. (2014). *MATLAB and Simulink software.* URL `http://www.mathworks.com [online]`.

Rodríguez, M. and Barrio, R. (2012). Reducing rounding errors and achieving brouwers law with taylor series method. *Applied Numerical Mathematics*, 62, 1014–1024.

Wolfram (2014). *MATHEMATICA software.* URL `http://www.wolfram.com/mathematica/ [online]`.