On computer algebra aided numerical solution of ODE by finite difference method

Yury A. Blinkov and Vladimir P. Gerdt

Except very special cases, nonlinear ordinary differential equations (ODE) admit numerical integration only. Historically first and one of the most-used numerical methods is finite difference method (FDM) [1] based on a finite difference approximation (FDA). As this takes place, the quality of numerical solution to PDE is determined by the quality of its FDA and by the method of numerical solution used to solve the difference equations comprising FDA.

One of the most challenging problems in to construct FDA which mimics basic algebraic properties of the ODE. Such mimetic FDA are more likely to produce highly accurate and stable numerical results (cf. [2]). In particular, a mimetic FDA is to be *totally conservative* (see [3], Def.1) what means the inheritance of algebraic integrals of the ODE at the discrete level.

Example. We consider the following autonomous ODE system [3]

$$\begin{cases} \dot{p} = qr, \\ \dot{q} = -pr, \qquad k = \text{const}\,, \\ \dot{r} = -k^2 pq \end{cases} \tag{1}$$

which has two quadratic integrals

$$p^2 + q^2 = \text{const} \quad \text{and} \quad k^2 p^2 + r^2 = \text{const}. \tag{2}$$

We use the denotations $\mathbf{x} := \{p, q, r\}$ and $\mathbf{F}(\mathbf{x}) := \{qr, -pr, -k^2 pq\}$ and consider the implicit midpoint finite difference discretization of system (1)

$$\frac{d\mathbf{x}}{dt} = \mathbf{F}(\mathbf{x}) \quad \Longrightarrow \quad \frac{\mathbf{x}_{n+1} - \mathbf{x}_n}{\Delta t} = \frac{\mathbf{F}(\mathbf{x}_{n+1}) + \mathbf{F}(\mathbf{x}_n)}{2} \tag{3}$$

It is known [4] that the scheme (3) preserves integrals (2). Instead of application of the Gröbner bases technique for solution of algebraic system (3) for transition to the next layer, as done in [3], we suggest computationally much more efficient the *simple iteration method*.

The work is supported in part by the Russian Foundation for Basic Research (grant No. 18-51-18005).

The construction of a simple iteration is based on the following quadratic formulas for monomials occurring in $\mathbf{F}(\mathbf{x})$

$$\begin{aligned} u^2 &= u^2 - u'^2 + u'^2 = (u-u')(u+u') + \\ &\qquad + u'^2 \approx (u-u')2u' + u'^2 = 2uu' - u'^2 \\ uv &= (u+v)^2 - (u-v)^2)/4 \approx ((2(u+v)(u'+v') - (u'+v')^2) - \\ &\quad - (2(u-v)(u'-v') - (u'-v')^2))/4 = uv' + u'v - u'v' \\ u^3 &= u^3 - u'^3 + u'^3 = (u-u')(u^2 + uu' + u'^2) + \\ &\qquad + u'^3 \approx (u-u')3u'^2 + u'^2 = 3uu'^2 - 2u'^3 \end{aligned} \tag{4}$$

and the qubic one

$$\begin{aligned} u^2v = uuv &\approx u'u'v + uu'v' + u'uv' - \\ &- (3-1)u'^2v' = u'^2v + 2uu'v' - 2u'^2v' \end{aligned} \tag{5}$$

The rule (5) can be easily implemented in a user's programming language of any modern computer algebra system and allows obtain the code for numerical computations. Thus, for the system (1) and its implicit scheme (3) the simple ireration method yields

$$\begin{cases} \dfrac{p_{n+1}-p_n}{\Delta t} - \dfrac{q_{n+1}r'_{n+1} + q'_{n+1}r_{n+1} - q'_{n+1}r'_{n+1} + q_nr_n}{2} = 0, \\ \dfrac{q_{n+1}-q_n}{\Delta t} - \dfrac{-p_{n+1}r'_{n+1} - p'_{n+1}r_{n+1} + p'_{n+1}r'_{n+1} - p_nr_n}{2} = 0, \\ \dfrac{r_{n+1}-r_n}{\Delta t} - \dfrac{-k^2p_{n+1}q'_{n+1} - k^2p'_{n+1}q_{n+1} + k^2p'_{n+1}q'_{n+1} - k^2p_nq_n}{2} = 0 \end{cases} \tag{6}$$

In the matrix form obtained by using the computer algebra system *SymPy* (`https://www.sympy.org`) is given by

$$\begin{aligned} &\begin{pmatrix} 1/\Delta t & -r'_{n+1}/2 & -q'_{n+1}/2 \\ r'_{n+1}/2 & 1/\Delta t & p'_{n+1}/2 \\ k^2q'_{n+1}/2 & k^2p'_{n+1}/2 & 1/\Delta t \end{pmatrix} \begin{pmatrix} p_{n+1} \\ q_{n+1} \\ r_{n+1} \end{pmatrix} = \\ &\qquad = \begin{pmatrix} p_n/\Delta t - q'_{n+1}r'_{n+1}/2 + q_nr_n/2 \\ q_n/\Delta t + p'_{n+1}r'_{n+1}/2 - p_nr_n/2 \\ r_n/\Delta t + k^2p'_{n+1}q'_{n+1}/2 - k^2p_nq_n/2 \end{pmatrix} \end{aligned} \tag{7}$$

The numerical results obtained with *SciPy* (`https://www.scipy.org/`) and illustrated by Fig. 1 show that there is no accumulation of numerical error, as distinguished from the output of standard ODE solvers `lsoda`, `vode`, `dopri5`, `dop853` of *SciPy*.

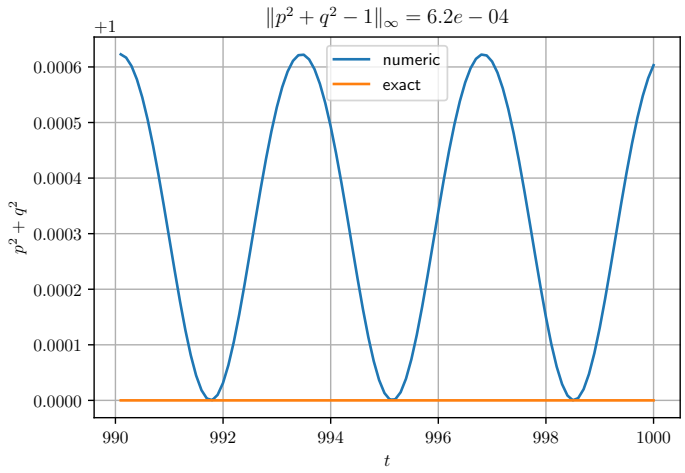

FIGURE 1. Dynamics of numerical error for $p = 0$, $q = r = 1$ and $k = 1/2$

References

[1] A. A. Samarskii. *Theory of Difference Schemes.* Marcel Dekker, New York, 2001.

[2] B. Koren, R. Abgral, P. Bochev, J. Frank and B. Perot (eds.) *Physics - compatible numerical methods.* J. Comput. Phys., 257, Part B, 1039–1526, 2014.

[3] Edik A. Ayryan, Mikhail D. Malykh, Leonid A. Sevastianov, Yu Ying. *Finite Difference Schemes and Classical Transcendental Functions* Lecture Notes in Computer Science, Vol. 11189, pp. 235-242, 2018.

[4] J. M. Sanz-Serna. *Symplectic Runge-Kutta schemes for adjoint equations, automatic differentiation, optimal control, and more.* SIAM Rev. 58(1), 3–33., 2016.

Yury A. Blinkov
Saratov State University
Saratov, Russia
e-mail: `blinkovua@info.sgu.ru`

Vladimir P. Gerdt
Joint Institute for Nuclear Research
Dubna, Russia
e-mail: `gerdt@jinr.ru`