Applied Mathematics Letters 43 (2015) 101–107

Contents lists available at ScienceDirect

Applied Mathematics Letters

journal homepage: www.elsevier.com/locate/aml

Accurate dense output formula for exponential integrators using the scaling and squaring method

Chengshan Wang [a], Xiaopeng Fu [a], Peng Li [a,*], Jianzhong Wu [b]

[a] Key Laboratory of Smart Grid of Ministry of Education, Tianjin University, Tianjin, China

[b] Institute of Energy, School of Engineering, Cardiff University, Cardiff, UK

ARTICLE INFO

Article history:
Received 9 October 2014
Received in revised form 7 December 2014
Accepted 8 December 2014
Available online 17 December 2014

Keywords:
Exponential integrator
Dense output formula
Matrix exponential

ABSTRACT

This paper proposes an accurate dense output formula for exponential integrators. The computation of matrix exponential function is a vital step in implementing exponential integrators. By scrutinizing the computational process of matrix exponentials using the scaling and squaring method, valuable intermediate results in this process are identified and then used to establish a dense output formula. Efficient computation of dense outputs by the proposed formula enables time integration methods to set their simulation step sizes more flexibly. The efficacy of the proposed formula is verified through numerical examples from the power engineering field.

1. Introduction

Exponential integrators are a class of time integration methods for differential equations and are especially suitable for stiff equations [1]. For the initial value problems

$$\boldsymbol{u}'(t) = \boldsymbol{F}(\boldsymbol{u}(t)), \qquad \boldsymbol{u}(t_0) = \boldsymbol{u}_0, \tag{1}$$

these methods perform a linearization procedure on the original problem,

$$\boldsymbol{u}'(t) = \boldsymbol{F}(\boldsymbol{u}(t)) = \boldsymbol{A}\boldsymbol{u}(t) + \boldsymbol{g}(\boldsymbol{u}(t)), \tag{2}$$

to extract a matrix $\boldsymbol{A}$ that retains the stiffness property. The associated matrix exponential and φ functions are embedded into integration schemes, and the nonlinear remainder $\boldsymbol{g}(\boldsymbol{u}(t))$ is then treated approximately. The φ functions are defined as follows [1,2]:

$$\varphi_0(z) = \mathrm{e}^z, \tag{3}$$

$$\varphi_k(z) = z\varphi_{k+1}(z) + 1/k!. \tag{4}$$

For the linear differential equations with polynomial inhomogeneity (as a truncated Taylor expansion of (2)),

$$\boldsymbol{u}'(t) = \boldsymbol{A}\boldsymbol{u}(t) + \sum_{k=0}^{p} \boldsymbol{g}_k/k!\,(t-t_0)^k, \qquad \boldsymbol{u}(t_0) = \boldsymbol{u}_0, \tag{5}$$

* Corresponding author.
E-mail addresses: cswang@tju.edu.cn (C. Wang), fuxiaopeng@tju.edu.cn (X. Fu), lip@tju.edu.cn (P. Li), wuj5@cardiff.ac.uk (J. Wu).

http://dx.doi.org/10.1016/j.aml.2014.12.008
0893-9659/

their solution can be expressed exactly using the matrix exponential and the φ functions [2], as follows:

$$\boldsymbol{u}(t) = \mathbf{e}^{(t-t_0)\boldsymbol{A}}\boldsymbol{u}_0 + \sum_{k=0}^{p}(t-t_0)^{k+1}\boldsymbol{\varphi}_{k+1}((t-t_0)\boldsymbol{A})\,\boldsymbol{g}_k. \tag{6}$$

The introduction of the matrix exponential and φ functions into the integration schemes allows for the analogous linear problem (5) to be analytically solved, which thus provides the exponential integrators with better numerical performance over that of traditional methods. However, the computational burden per step also grows. In spite of breakthroughs that have been achieved in recent years [3–5], matrix exponential-related functions are more computationally intense compared to the solution of a linear system that has the same dimension. For applications in the engineering fields, certain supplements to the algorithmic implementation are required to fit the intrinsic properties of exponential integrators to the actual demands of engineering analysis. The development of the dense output formula is one possible approach to facilitate engineering applications of exponential integrators.

In engineering applications, the choice of the time integration step size involves consideration of both the numerical requirements, i.e., the deviation of the solution at selected time discretization nodes, and the non-numerical constraints, such as visualization requirements for the simulation results. Since simulation is typically not the final stage of the entire analysis process, the simulation result itself is an input to subsequent analysis stages ("advanced applications"), such as in a frequency spectrum scan, which often demands a certain density in time discretization. In this case, for methods that offer high accuracy, such as exponential integrators, if the step size is chosen solely from a numerical perspective, an insufficient sampling frequency is provided; if a smaller step size is adopted due to external constraints, then the computational cost is high and it is difficult to compete with low-order methods. Taking power system engineering as an example, the dominant time integration method in this field is the implicit trapezoidal rule [6]. More advanced and recent numerical methods are seldom used.

The dense output formula decouples the output step size from the computational step size, which therefore provides flexibility in addressing the above-mentioned dilemma. For an interval $[t_n,\ t_{n+1}]$ that is formed by consecutive time discretization nodes (which is now chosen only for the numerical aspect), a dense output formula provides cheap outputs on $\{t_{n,k}|k = 0, 1, \ldots, \sigma$, s.t. $t_{n,0} = t_n, t_{n,\sigma} = t_{n+1}, t_{n,k+1} - t_{n,k} = (t_{n+1} - t_n)/\sigma\}$. Going one step further, if the accuracy of these dense outputs does not deteriorate the overall simulation, it can be seen that the simulation step size shrinks to $1/\sigma$, and the per-step computational cost is spread to these σ small steps. This action equivalently boosts the efficiency of the exponential integrators and enables them to gain an edge in computational speed over low-order methods. Dense output is also important in addressing other practical issues, such as event location and the treatment of discontinuities in the simulation.

In this paper, a dense output formula that is embedded into the matrix exponential computational process is proposed. Unlike other generic approaches, e.g., interpolation, our approach is tailored for exponential integrators and fully utilizes the peculiar properties of the matrix exponential function. Our approach has a clear physical meaning and an efficient implementation, and it can accurately recover the oscillatory dynamics between time discretization nodes. One of the most widely used exponential integrators, the Exponential Rosenbrock–Euler (ERE) method [7,8], was chosen to demonstrate the proposed approach.

2. Core steps of exponential integrators: ERE method as an example

The ERE method is a Rosenbrock-type exponential integrator with a stiff order of 2 [7,8]. For the general nonlinear system (1), the ERE method performs continuous linearization along the numerical solution at each step, i.e.,

$$\boldsymbol{u}'(t) = \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}}(\boldsymbol{u}_n)\,\boldsymbol{u}(t) + \left[\boldsymbol{F}(\boldsymbol{u}(t)) - \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}}(\boldsymbol{u}_n)\,\boldsymbol{u}(t)\right] := \boldsymbol{A}_n\boldsymbol{u}(t) + \boldsymbol{g}_n(\boldsymbol{u}(t)), \tag{7}$$

where $\boldsymbol{u}_n \in \mathbb{R}^N$ is the numerical approximation of the accurate $\boldsymbol{u}(t_n)$. Using the variation-of-constant formula [1,2], the following matrix exponential function is introduced:

$$\boldsymbol{u}(t) = \mathbf{e}^{(t-t_n)\boldsymbol{A}_n}\boldsymbol{u}(t_n) + \int_{t_n}^{t}\mathbf{e}^{(t-\tau)\boldsymbol{A}_n}\boldsymbol{g}_n(\boldsymbol{u}(\tau))\,d\tau. \tag{8}$$

For simplicity, $\boldsymbol{g}_n(\boldsymbol{u}(\tau))$ is approximated with $\boldsymbol{g}_n(\boldsymbol{u}_n)$ over the interval $[t_n, t_{n+1}]$. Let $h_n = t_{n+1} - t_n$, the ERE integration scheme is derived as follows:

$$\boldsymbol{u}_{n+1} = \mathbf{e}^{h_n\boldsymbol{A}_n}\boldsymbol{u}_n + h_n\boldsymbol{\varphi}_1(h_n\boldsymbol{A}_n)\,\boldsymbol{g}_n(\boldsymbol{u}_n). \tag{9}$$

In practice, a separate computation for $\mathbf{e}^{h_n\boldsymbol{A}_n}$ and $\boldsymbol{\varphi}_1(h_n\boldsymbol{A}_n)$ is not required. Using a technique from [9], the computation of $\boldsymbol{\varphi}_1(h_n\boldsymbol{A}_n)$ is absorbed into a larger matrix exponential computation. The final integration formula is as follows:

$$\boldsymbol{u}(t_{n+1}) \approx \boldsymbol{u}_{n+1} = \begin{bmatrix}\mathbf{I}_N & \mathbf{0}_{N\times 1}\end{bmatrix}\mathbf{e}^{h_n\begin{bmatrix}\boldsymbol{A}_n & \boldsymbol{g}_n(\boldsymbol{u}_n)\\ \mathbf{0}_{1\times N} & 0\end{bmatrix}}\begin{bmatrix}\boldsymbol{u}_n\\ 1\end{bmatrix} := \begin{bmatrix}\mathbf{I}_N & \mathbf{0}_{N\times 1}\end{bmatrix}\mathbf{e}^{h_n\tilde{\boldsymbol{A}}_n}\tilde{\boldsymbol{u}}_n. \tag{10}$$

Note that matrix $\begin{bmatrix}\mathbf{I}_N & \mathbf{0}_{N\times 1}\end{bmatrix}$ is a vector slicing operator that does not imply real matrix–vector multiplication.