Letters

A Graph Propagation-Based Automatic Differentiation Method for Power Electronic and Electric Machine Systems Simulation

Shengyu Jia, *Graduate Student Member, IEEE*, Bochen Shi, *Member, IEEE*, Zhengming Zhao, *Fellow, IEEE*, and Liqiang Yuan, *Senior Member, IEEE*

***Abstract*—This letter proposes an automatic differential method to enable the effective Taylor-series based flexible integration algorithm for power electronics and electric machine systems simulation. The proposed approach decomposes the system equations into linear and nonlinear parts and encapsulates the nonlinear part through controlled-source interfaces. Furthermore, this letter introduces a graph propagation-based automatic differentiation method (GP-AD), which enables the automatic generation of higher order derivatives for nonlinear portion. To evaluate the efficiency of the proposed approach, a case study on a power traction system of a high-speed train is presented. The results indicate that the proposed algorithm achieves a sixfold speedup compared to the commercial software, demonstrating the effectiveness of the proposed method.**

***Index Terms*—Adaptive taylor series based integration, automated differentiation, higher-order derivatives, nonlinear, power electronics.**

I. Introduction

MODERN electric machine (EM) drive systems with power electronics (PE) devices often exhibit a mixed structure that combines linear and nonlinear components [1], which poses challenges for existing simulation integration algorithms.

Various algorithms can solve nonlinear ordinary differential equations (ODEs), but they often struggle with inefficiencies in PE simulations due to limited flexibility [2]. Among methods leveraging linear properties for acceleration [3], [4], [5], the Taylor series-based flexible adaptive (FA) algorithm [4] dynamically adjusts integration step size and order over varying time intervals, making it particularly effective for PE systems simulation.

However, extending the FA algorithm to PE-EM systems poses significant challenges in obtaining higher order derivatives for nonlinear equations [6]. Manual symbolic differentiation of nonlinear machine equations is complex and error-prone [7]. While automatic differentiation (AD) techniques have been proposed, existing implementations have limitations. Numerical differentiation methods [8], [9] suffer from numerical errors and restricted derivative orders due to error accumulation. Symbolic expression-based AD tools [10] offer high precision but incur high-computational costs, making them inefficient for PE-EM simulations.

Thus, this letter aims to propose an efficient and accurate automatic higher order derivative computation method, enabling the FA algorithm's automatic implementation in PE-EM systems. The approach decomposes system equations into linear and nonlinear components, encapsulating nonlinearity through controlled sources. The proposed graph propagation-based automatic differentiation (GP-AD) method represents nonlinear equations as a directed graph (DG), propagating derivatives along graph paths via the chain rule. By integrating GP-AD, the FA algorithm adaptively computes higher order derivatives for state and algebraic variables while adjusting step size and integration order through error control, ensuring high-efficiency simulation for PE-EM systems.

II. System Decoupled Modeling for Linear and Nonlinear Subsystems With Algebraic Interfaces

Generally, the system can be represented with the state space equations as follows:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \tag{1a}$$

$$\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}) \tag{1b}$$

where $\boldsymbol{x} \in R^{n\times 1}$ denotes the state variables, $\boldsymbol{u} \in R^{m\times 1}$ includes inputs such as voltage and current source. $\boldsymbol{y} \in R^{l\times 1}$ denotes the output vector. $\boldsymbol{f} : R^{n\times 1} \times R^{m\times 1} \to R^{n\times 1}$ represents the state equations, and $\boldsymbol{g} : R^{n\times 1} \times R^{m\times 1} \to R^{l\times 1}$ denotes the output equations.

For linear systems, the function $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ can typically be expressed in the form $\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$, where $\boldsymbol{A}$ and $\boldsymbol{B}$ are system coefficient matrices. However, for nonlinear components such as EMs, the presence of factors such as electromagnetic torque and magnetic saturation introduces nonadditive and nonlinear behaviors in the state equations. As a result, these equations

Received 12 March 2025; revised 6 May 2025; accepted 19 May 2025. Date of publication 27 May 2025; date of current version 5 August 2025. This work was supported in part by the National Key R&D Program of China under Grant 2023YFB3307000 and in part by the National Natural Science Foundation of China under Grant 52307211. *(Corresponding author: Bochen Shi.)*

The authors are with the Department of Electrical Engineering, Tsinghua University, Beijing 100084, China (e-mail: jsy20@mails.tsinghua.edu.cn; sbc@mail.tsinghua.edu.cn; zhaozm@tsinghua.edu.cn; ylq@tsinghua.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TPEL.2025.3574339.

Digital Object Identifier 10.1109/TPEL.2025.3574339

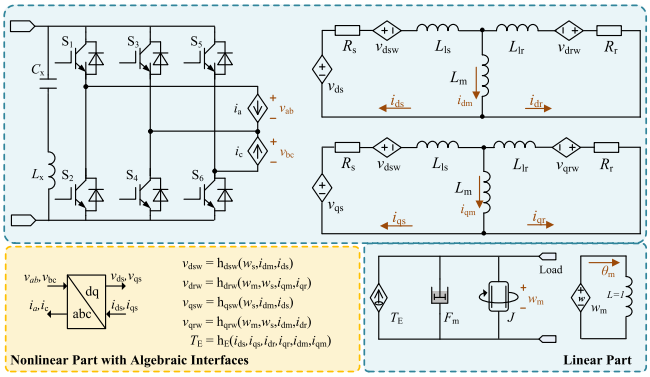

Fig. 1. Schematic of a motor drive system, illustrating the decomposition into a linear subsystem and algebraic nonlinear interfaces.

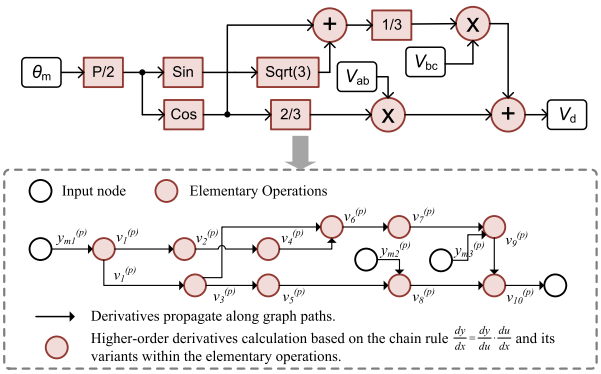

Fig. 2. GP-AD diagram.

cannot be represented in linear form, which accounts for the increased complexity in computing higher order derivatives.

Although machine equations appear to be nonlinear overall, it can be separated to the linear and nonlinear parts by introducing controlled sources in the equivalent circuit modeling. For instance, as shown in Fig. 1, the dq-axis equivalent circuit of an induction machine primarily consists of *RLC* components, which can be treated as linear. The nonlinear effects, including electromagnetic torque, back-EMF, and dq transformation, are incorporated into algebraic equations through controlled sources and measurement outputs as controlling quantities.

Thus, the controlled source acts as an "algebraic interface" linking the linear and nonlinear parts. Each interface is defined by an output $\boldsymbol{u}_{\mathrm{c}}$ driven by measured signals $\boldsymbol{y}_{\mathrm{m}}$ as follows:

$$\boldsymbol{u}_{\mathrm{c}} = \boldsymbol{h}\left(\boldsymbol{y}_{\mathrm{m}}\right) \tag{2}$$

where $\boldsymbol{h}(\cdot)$ is constructed from elementary operations (e.g., addition, multiplication, and trigonometric functions).

This modeling technique effectively splits the machine's nonlinear equations by placing all differential equations in the linear part and only algebraic equations in the nonlinear portion. Then, the model can be expressed as (3) and (4).

$$\dot{\boldsymbol{x}} = \boldsymbol{A}_k\,\boldsymbol{x} + \begin{bmatrix}\boldsymbol{B}_k^1 & \boldsymbol{B}_k^2\end{bmatrix}\begin{bmatrix}\boldsymbol{u}_{\mathrm{e}}\\ \boldsymbol{u}_{\mathrm{c}}\end{bmatrix} \tag{3a}$$

$$\boldsymbol{y} = \boldsymbol{C}_k\,\boldsymbol{x} + \begin{bmatrix}\boldsymbol{D}_k^1 & \boldsymbol{D}_k^2\end{bmatrix}\begin{bmatrix}\boldsymbol{u}_{\mathrm{e}}\\ \boldsymbol{u}_{\mathrm{c}}\end{bmatrix} \tag{3b}$$

with $\boldsymbol{u}_{\mathrm{c}}$ determined as follows:

$$\boldsymbol{u}_{\mathrm{c}} = \boldsymbol{h}_{\mathrm{nl}}(\boldsymbol{y}_{\mathrm{m}}) \tag{4a}$$

$$\boldsymbol{y}_{\mathrm{m}} = \boldsymbol{C}_{mk}\,\boldsymbol{x} + \boldsymbol{D}_{mk}\,\boldsymbol{u}_{\mathrm{e}} \tag{4b}$$

where (3) denotes the linear part, and (4) denotes the nonlinear part. $\boldsymbol{u}_{\mathrm{c}} \in R^{m_c\times 1}$ denotes the controlled source (as the role of algebraic interface) and $\boldsymbol{u}_{\mathrm{e}} \in R^{m_e\times 1}$ denotes the independent input. $\boldsymbol{y}_{\mathrm{m}} \in R^{l_m\times 1}$ is the measurement output associated with $\boldsymbol{u}_{\mathrm{c}}$. $\boldsymbol{A}_k, \boldsymbol{B}_k, \boldsymbol{C}_k, \boldsymbol{D}_k$ are matrices determined by circuit topology and parameters, and k denotes different topologies decided by switch states. $\boldsymbol{h}_{\mathrm{nl}} : R^{l_m\times 1} \to R^{m_c\times 1}$ represents the nonlinear equations in the system. $\boldsymbol{C}_{mk}, \boldsymbol{D}_{mk}$ are matrices related to calculation of $\boldsymbol{y}_{\mathrm{m}}$.

This modeling technique separates the PE-EM system's linear and nonlinear equations using an algebraic interface, enabling independent treatment of the two parts. This provides the foundation for the subsequent algorithm implementation.

III. GP-AD and FA Integration Algorithm

Before introducing the specific method, the overall solution procedure for the system equations is first briefly outlined. By decoupling the system equations as expressed in (3) and (4), a semiexplicit differential algebraic equation (DAE) [11] formulation is obtained. This allows a stepwise solution strategy at each time step t_n as follows.

1) Solve (3) to obtain $\boldsymbol{x}$ at t_{n+1} with order $= p$; and update the needed measurement output $\boldsymbol{y}_m$ of pth order with (4b);
2) Solve the nonlinear part (4a) (i.e., algebraic interface) to update pth-order $\boldsymbol{u}_{\mathrm{c}}$ at t_{n+1}.

Building on this foundation, this section then presents the GP-AD method and its integration into the FA algorithm.

A. Nonlinear Modeling With Elementary Operations to Form the DG

To compute higher order derivatives of the algebraic interface u_c, we first derive the measurement vector y_m's derivatives by differentiating (4b) as follows:

$$\boldsymbol{y}_{\mathrm{m}}^{(i)}(t) = \boldsymbol{C}_{mk}\,\boldsymbol{x}^{(i)}(t) + \boldsymbol{D}_{mk}^1\,\boldsymbol{u}_{\mathrm{e}}^{(i)}(t). \tag{5}$$

Since (4a) defines $\boldsymbol{u}_{\mathrm{c}}$ as a function of $\boldsymbol{y}_{\mathrm{m}}$, once $\boldsymbol{y}_{\mathrm{m}}^{(i)}$ is obtained, it can be substituted into $\boldsymbol{h}$ to compute the corresponding derivative of $\boldsymbol{u}_{\mathrm{c}}$.

Since the function $\boldsymbol{h}_{\mathrm{nl}}(\cdot)$ is composed of elementary operations, it can be represented as a computation graph consisting of these operations. The derivatives can then be computed locally for each operation and subsequently combined to obtain the full derivative. By structuring $\boldsymbol{h}$ in this way, an elementwise DG is constructed, enabling derivative propagation from $\boldsymbol{y}_m$ to $\boldsymbol{u}_c$. This modeling approach is illustrated in Fig. 2, where the red elements denote individual elementary operations.

The directed graph is denoted as $G(V, E)$. The nodes V of this graph consist of the input variables $\boldsymbol{y}_m$, the output variables

$\boldsymbol{u}_c$, and intermediate computational nodes $h_i(\cdot)$, while the edges E represent the data flow between these nodes. The depth-first search algorithm [12] is used to determine the computational dependencies by identifying paths from output nodes $\boldsymbol{u}_c$ back to input nodes $\boldsymbol{y}_m$, generating computational chains $\mathcal{P} = P_i$, where each path P_i represents a sequence of operations.

B. Automatic Derivative Via Graph Propagation

Once the computational paths $\mathcal{P}$ are identified, higher order derivatives of $\boldsymbol{u}_c$ can be propagated along these paths. Let v_j represent the output of a node along a path $P_i \in \mathcal{P}$, and let h_j represent the operation associated with v_j. For single-input functions (e.g., $\cos, \exp$), the qth-order derivative follows the Faà di Bruno formula [13], which is a generalizing of chain rule to higher derivatives as follows:

$$\frac{d^q v_j}{dt^q} = \sum_{r=1}^{q} h^{(r)}(\boldsymbol{v}_{j-1})\, B_{q,r}\left(\frac{dv_{j-1}}{dt}, \ldots, \frac{d^{q-r+1}v_{j-1}}{dt^{q-r+1}}\right) \quad (6)$$

where $B_{q,r}$ are the partial exponential Bell polynomials, which determine the composition of the higher order derivatives of v_{j-1}.

For multi-input functions (e.g., multiplication, division), following the chain rule, the qth-order derivative is as follows:

$$\frac{d^q v_j}{dt^q} = \sum_{r=0}^{q} C_{q,r} \prod_{k=1}^{m_j} \frac{d^{q_{r,k}} v_{j,in,k}}{dt^{q_{r,k}}} \quad (7)$$

where $v_{j,\text{in},k}$ denotes the kth input node to v_j, and m_j is the number of inputs to v_j. The term $C_{q,r}$ represents the combinatorial coefficient for the rth term in the computation of the qth derivative (for the case of two inputs, it corresponds to the binomial coefficient). The quantity $q_{r,k}$ denotes the order of differentiation for the kth input in the rth term, satisfying $\sum_{k=1}^{m_j} q_{r,k} = q$.

For a specific computational path $P_i = \{\boldsymbol{y}_m = v_0, v_1, v_2, \ldots, v_n, \boldsymbol{u}_c = v_{n+1}\}$, the complete derivative of $\boldsymbol{u}_c$ is obtained by appropriately aggregating the derivatives along all paths $P_i \in \mathcal{P}$. The elementary operators are precomputed in a library along with their high-order partial derivatives. In addition, the previously computed derivatives are stored and reused to reduce computational redundancy. Overall, the GP-AD method is shown in Fig. 2.

C. FA Integration Algorithm and Whole Flowchart

Suppose that for the state variable $\boldsymbol{x}$, all derivatives at t_{n-1} are known. Based on the Taylor series expansion, the numerical solution of the state equation at time t_n is as follows:

$$\boldsymbol{x}_n = \boldsymbol{x}_{n-1} + \sum_{i=1}^{p} \frac{\boldsymbol{x}_{n-1}^{(i)}}{i!} (\Delta t_n)^i \quad (8)$$

where Δt_n is the step size for the nth step, $\boldsymbol{x}_{n-1}^{(i)}$ is the ith time derivative of $\boldsymbol{x}$ at t_{n-1}, and p is the order at which the series is truncated.

The local truncation error (LTE) at t_n is the sum of higher order terms beyond p and can be approximated by its leading term (the $(p+1)$th-order) as follows:

$$\varepsilon_n \approx \frac{\boldsymbol{x}_{n-1}^{(p+1)}}{(p+1)!} (\Delta t_n)^{p+1}. \quad (9)$$

Thus, the FA algorithm has an LTE of $O(p+1)$.

Bringing the FA integration algorithm and the GP-AD technique together, the overall flowchart of solving the PE-EM systems is shown in Fig. 3.

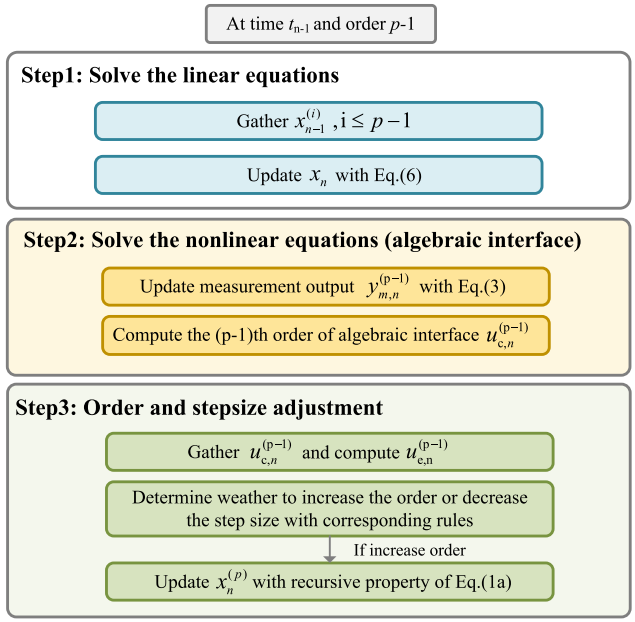

Fig. 3. Flowchart of the FA integration algorithm with the proposed GP-AD approach.

D. Algorithm Error and Numerical Stability Analysis

We first analyze the LTE, which consists of two components: the integration error from the FA algorithm and the differentiation error from the GP-AD process. The LTE of the pth-order FA method is given by (9). Defining the numerical error as $\boldsymbol{x}_{\text{num}} - \boldsymbol{x}_{\text{exact}} = \boldsymbol{\varepsilon}_x = O(\Delta t_n^{p+1})$, where the subscripts num and exact denote the numerical and exact solutions, respectively. Then, considering the proposed GP-AD approach, the LTE can be derived as follows.

From the measurement calculation (4b), the measurement error is as follows:

$$\Delta \boldsymbol{y}_m = \boldsymbol{y}_{m,\text{num}} - \boldsymbol{y}_{m,\text{exact}} = \boldsymbol{C}_{mk}\, \boldsymbol{\varepsilon}_x. \quad (10)$$

Since $\boldsymbol{h}$ is a smooth function, its Taylor series expansion around $\boldsymbol{y}_{m,\text{exact}}$ gives

$$\boldsymbol{h}(\boldsymbol{y}_{m,\text{exact}} + \Delta \boldsymbol{y}_m) = \boldsymbol{h}(\boldsymbol{y}_{m,\text{exact}}) + \boldsymbol{h}'(\boldsymbol{y}_{m,\text{exact}})\, \Delta \boldsymbol{y}_m + O(\|\Delta \boldsymbol{y}_m\|^2). \quad (11)$$

The algebraic interface error is defined as follows:

$$\varepsilon_{\boldsymbol{u}_c} = \boldsymbol{u}_{c,\text{num}} - \boldsymbol{u}_{c,\text{exact}} = \boldsymbol{h}(\boldsymbol{y}_{m,\text{num}}) - \boldsymbol{h}(\boldsymbol{y}_{m,\text{exact}}). \quad (12)$$

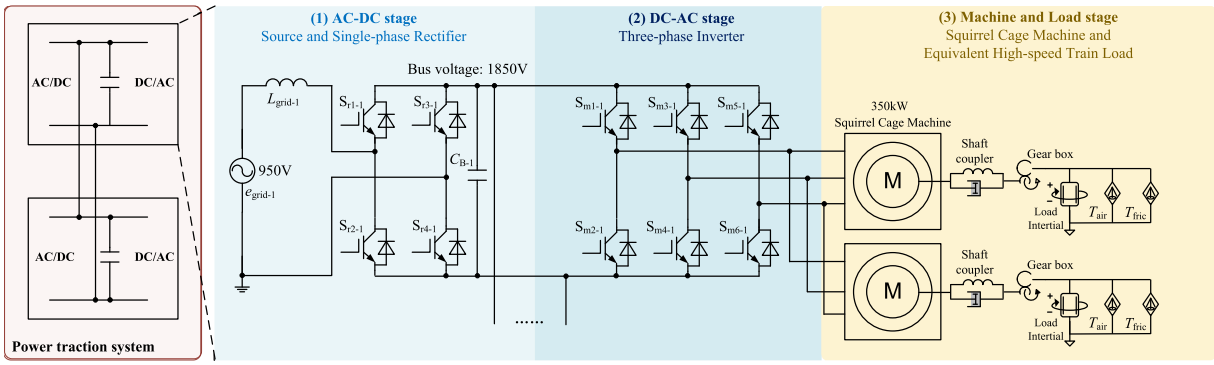

Fig. 4. Topology diagram of the traction converter driving the induction motor.

Substituting the Taylor expansion, we obtain

$$\boldsymbol{\varepsilon}_{\boldsymbol{u}_c} \approx \boldsymbol{h}'(\boldsymbol{y}_{m,\text{exact}})\,\Delta y_m = \boldsymbol{h}'(\boldsymbol{y}_{m,\text{exact}})\,\boldsymbol{C}_{mk}\,\boldsymbol{\varepsilon}_x. \tag{13}$$

Since $\boldsymbol{\varepsilon}_x = O(\Delta t_n^{p+1})$, it follows that

$$\boldsymbol{\varepsilon}_{u_c} = O(\Delta t_n^{p+1}). \tag{14}$$

Thus, both the linear and nonlinear interface maintain an error order of $p+1$ as follows:

$$\boldsymbol{\varepsilon}_{\text{total}} = \|\boldsymbol{x}_{\text{num}} - \boldsymbol{x}_{\text{exact}}\| + \|\boldsymbol{u}_{c,\text{num}} - \boldsymbol{u}_{c,\text{exact}}\| = O(h^{p+1}). \tag{15}$$

Second, regarding numerical stability, the proposed GP-AD method does not introduce additional errors and, therefore, does not negatively affect the overall numerical stability. The stability characteristics and stability region of the algorithm are primarily determined by the FA method, which shares the same stability properties as typical forward integration schemes [4].

IV. Case Study

To evaluate the simulation accuracy and efficiency of the proposed method, this section presents a case study based on the power traction system of a high-speed train. High-speed train traction is a representative and important industrial application of PE-EM systems. Typically, a high-speed train consists of multiple carriages, with each traction converter driving several induction motors—forming a typical PE-EM system that includes both linear and nonlinear components. This system configuration directly aligns with the target application scenarios addressed by the proposed method. Therefore, by analyzing the simulation performance of this case, the effectiveness and practicality of the proposed method can be validated.

The topology of the traction converter are shown in Fig. 4. The simulation captures a scenario where the train accelerates from rest to a medium speed (approximately 250 km/h), with a constant acceleration of 0.5 m/s^2. The motor shaft is connected to the load through a coupling and a gearbox. The load consists of three components: inertia load, mechanical resistance load, and aerodynamic resistance load. The key system parameters are listed in Table I.

Fig. 5 presents overall simulation results, including the rotor speed, mechanical load torque, electromagnetic torque, and dc-link voltage. To verify the accuracy, the proposed method is compared against results obtained from a commercial software. Furthermore, Fig. 6 compares the transient waveforms of different parts of the system at various dynamic moments. These include the stator flux and current during the start-up process or in a short period after steady state is reached, and the dynamic response of mechanical load torque and electromagnetic torque following the end of the acceleration phase.

TABLE I
Parameters of the High-Speed Train Traction System

Parameter	Value
Input AC voltage (RMS) U_a	950 V
DC bus voltage U_c	1850 V
Rated power per motor P_e	350 kW
Rated torque per motor T_e	800 Nm
Air resistance torque coefficient K_a	0.254
Mechanical resistance torque T_f	66 Nm
Load Rotational inertia	2600 kg·m²

Then, to quantitatively compare the results, absolute and relative error evaluations for several key quantities are presented. The detailed error comparison results are summarized in Tables II and III. In Table II, err_{rel} denotes the relative error, and err_{abs} denotes the absolute error. For metrics such as current THD and normalized ripple RMS, which have relatively small magnitudes, the comparisons are performed using absolute error. Other quantities are compared using either relative error. In Table III, the relative error for waveform time series was computed point-by-point over time, using the commercial software results as the reference.

Based on the above error analysis and waveform comparisons, it can be observed that the proposed method achieves a high degree of consistency with the commercial software for most waveform results. However, some discrepancies—with relative errors around 3%—can still be seen in certain cases, along with minor differences in transient waveforms. The main reason for these differences is that due to inherent differences between the two machine models, exact equivalence is difficult to achieve. Overall, when these results are viewed in the context of the full analysis and comparison, the differences remain within an

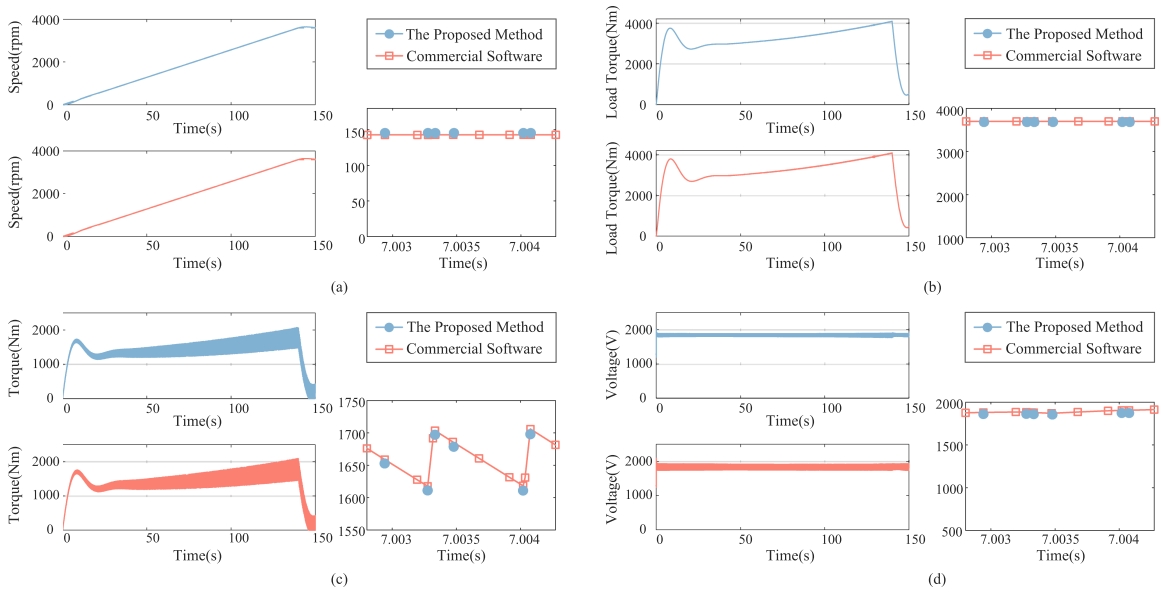

Fig. 5. Comparisons between the simulated results. (a) Rotor speed of the machine. (b) Torque of mechanical load. c) Electromagnetic torque output of the machine. (d) DC-link voltage.

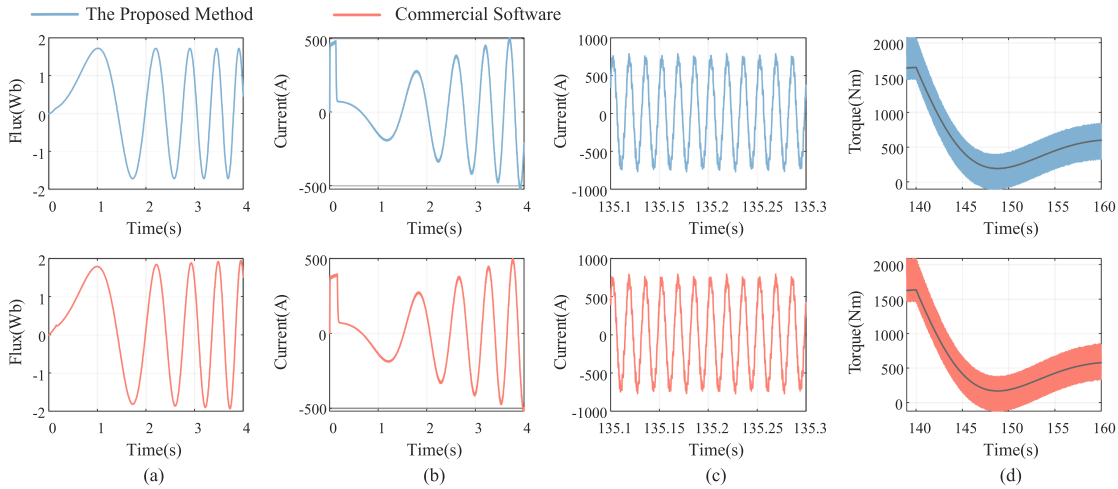

Fig. 6. Transient waveforms of different parts and scenarios. (a) Q-axis stator flux during startup process. (b) Phase A stator current during startup process. (c) Phase A stator current during a short transient period. (d) Electromagnetic torque and mechanical load torque (gray waveform) after acceleration termination.

TABLE II
COMPARISON OF CURRENT AND CONTROL PERFORMANCE

Method	Speed control peak error time	Speed control settling time	Load stabilization time (from 140s)	THD of A phase stator current	Normalized Ripple RMS
Proposed method	15.61 s	23.85 s	26.81 s	0.09%	2.80%
Commercial software	15.79 s	24.10 s	27.77 s	0.13%	2.44%
Absolute/Relative error	$err_{\text{rel}} = 1.14\%$	$err_{\text{rel}} = 1.07\%$	$err_{\text{rel}} = 3.46\%$	$err_{\text{abs}} = 0.04\%$	$err_{\text{abs}} = 0.36\%$

acceptable range, and the outcomes confirm the accuracy and validity of the proposed method.

Table IV provides a comparison of simulation efficiency among the proposed automated derivative-FA algorithm within the discrete state event driven (DSED) framework, the Dormand–Prince (RK45) algorithm within the DSED framework, and the RK45 algorithm implemented in commercial software. In this letter, piecewise linear electrical circuit simulation (PLECS) is employed as the commercial benchmark for simulation comparison. Within PLECS, the Dormand–Prince method (DOPRI) solver, which is essentially an RK45 algorithm, is used for numerical integration. The relative error setting of the solvers are all 1e-6. All three methods simulate a 150 s dynamic process.

TABLE III
OVERALL RELATIVE ERROR

Variable	Relative Error
Electromagnetic torque	3.42%
Mechanical load	0.94%
Rotational speed	0.74%
Time of switching actions	0.15%

TABLE IV
PERFORMANCE COMPARISON OF DIFFERENT ALGORITHMS

Metric	DSED/FA	DSED/RK45	Commercial Software/RK45
Total CPU time (s)	59.24	187.20	354.26
Normalized CPU time	1	3.16	5.98
Number of computation points	9.02×10^5	1.06×10^6	2.58×10^6
CPU time per step (us)	65.68	176.60	137.31
Total time for solving nonlinear part (s)	18.97	56.15	\

We briefly note that in the DSED framework, a semisymbolic modeling approach [14] is adopted to represent switching devices, allowing the coefficient matrices to be updated dynamically as the switching states change. As this work primarily focuses on the overall system-level behavior, switching transients are not explicitly modeled. However, if detailed transient modeling is required, the piecewise analytical transient model [15], [16] can be employed to efficiently simulate the switching process.

It can be observed from Table IV that the proposed method achieves a sixfold speedup compared to the commercial software. The table also provides a breakdown of the simulation time, including the total integration time and the time spent computing the nonlinear algebraic interface within the proposed method. The results indicate that the computational cost for nonlinear part accounts for only about $1/3$ of the total simulation time. This further demonstrates the effectiveness of the proposed approach.

V. CONCLUSION

This letter proposes a GP-AD method integrated with a FA integration algorithm for simulating PE and EM systems. GP-AD automates higher order derivative computation using graph propagation and chain rules, with precomputed expressions enhancing efficiency. Combined with the variable-step, variable-order FA algorithm and separately handling the linear and nonlinear part, it ensures accurate and efficient numerical integration. A case study on a high-speed train traction system shows a sixfold speedup over commercial simulation software while maintaining accuracy, demonstrating the method's effectiveness.

In the following, we discuss the computational efficiency and practical application scenarios of the proposed method. The GP-AD method, combined with the FA integration algorithm, enhances simulation efficiency by adaptively adjusting the integration order in systems with frequent discrete events. Since the FA algorithm is essentially a forward integration method, the proposed approach demonstrates significant acceleration benefits in the simulation of nonstiff systems. These benefits are particularly prominent in large-scale circuits with frequent switching events, where the ability to flexibly vary the integration order enables better adaptation to the system's dynamic behavior and variable step-size constraints. From an industrial perspective, the proposed method is particularly applicable to scenarios involving PE and EM systems, as well as components such as photovoltaics and batteries. It is also well-suited for modeling systems where magnetic saturation effects must be considered, such as in EMs and transformers.

REFERENCES

[1] A. Medina, A. Ramos-Paz, and C. Fuerte-Esquivel, "Periodic steady state solution of electric systems with nonlinear components using parallel processing," *IEEE Trans. Power Syst.*, vol. 18, no. 2, pp. 963–965, May 2003.
[2] L. Petzold, "Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations," *SIAM J. Sci. Stat. Comput.*, vol. 4, no. 1, Jul. 2006, Art. no. 0904010.
[3] C.-C. Liu, J. Hsieh, C. Chang, J. Bocek, and Y.-T. Hsiao, "A fast-decoupled method for time-domain simulation of power converters," *IEEE Trans. Power Electron.*, vol. 8, no. 1, pp. 37–45, Jan. 1993.
[4] Y. Zhu, Z. Zhao, B. Shi, and Z. Yu, "Discrete state event-driven framework with a flexible adaptive algorithm for simulation of power electronic systems," *IEEE Trans. Power Electron.*, vol. 34, no. 12, pp. 11692–11705, Dec. 2019.
[5] V. Satek, P. Veigend, and G. Necasova, "Taylor series based integration in electric circuits simulations," *Adv. Elect. Electron. Eng.*, vol. 17, no. 3, pp. 352–359, Sep. 2019.
[6] A. Ghorbani and M. Gachpazan, "A high-order algorithm for solving nonlinear algebraic equations," *Iranian J. Numer. Anal. Optim.*, vol. 11, no. 1, pp. 107–115, Mar. 2021.
[7] Y. Ma, V. Dixit, M. J. Innes, X. Guo, and C. Rackauckas, "A comparison of automatic differentiation and continuous sensitivity analysis for derivatives of differential equation solutions," in *Proc. 2021 IEEE High Perform. Extreme Comput. Conf. (HPEC)*. Waltham, MA, USA: IEEE, Sep. 2021, pp. 1–9.
[8] H. Xu, B. Shi, Z. Yu, J. Zheng, and Z. Zhao, "Numerical derivative-based flexible integration algorithm for power electronic systems simulation considering nonlinear components," *IEEE Trans. Ind. Electron.*, vol. 71, no. 9, pp. 10761-10771, Sep. 2024.
[9] E. Miletics and G. Molnárka, "Taylor series method with numerical derivatives for initial value problems," *J. Comput. Methods Sci. Eng.*, vol. 4, no. 1–2, pp. 105–114, Jan. 2004.
[10] U. Schmitt, B. Moser, C. S. Lorenz, and A. Réfrégier, "Sympy2c: From symbolic expressions to fast c/c functions and ODE solvers in python," *Astron. Comput.*, vol. 42, Jan. 2023, Art. no. 100666.
[11] M. Arnold, K. Strehmel, and R. Weiner, "Half-explicit Runge-Kutta methods for semi-explicit differential-algebraic equations of index 1," *Numerische Mathematik*, vol. 64, no. 1, pp. 409–431, Dec. 1993.
[12] S. Even, *Graph Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, Sep. 2011.
[13] W. P. Johnson, "The curious history of faà di bruno's formula," *The Amer. Math. Monthly*, vol. 109, no. 3, pp. 217–234, Mar. 2002.
[14] Z. Yu, Z. Zhao, B. Shi, Y. Zhu, and J. Ju, "An automated semi–symbolic state equation generation method for simulation of power electronic systems," *IEEE Trans. Power Electron.*, vol. 36, no. 4, pp. 3946–3956, Apr. 2021.
[15] B. Shi, Z. Zhao, and Y. Zhu, "Piecewise analytical transient model for power switching device commutation unit," *IEEE Trans. Power Electron.*, vol. 34, no. 6, pp. 5720–5736, Jun. 2019.
[16] Y. Xiao, Z. Zhao, B. Shi, Z. Yu, S. Jia, and S. Ji, "Piecewise analytical transient model of SiC MOSFET and SiC Schottky diode pair," in *Proc. IEEE 24th Workshop Control Model. Power Electron. (COMPEL)*. Ann Arbor, MI, USA: IEEE, Jun. 2023, pp. 1–6.