American Journal of Civil Engineering
2025, Vol. 13, No. 5, pp. 304-312
https://doi.org/10.11648/j.ajce.20251305.15

Research Article

Benchmarking Numerical Solvers for Damped Single-Degree-of-Freedom Vibration Systems: Technical Evaluation and Educational Insights

John Cannon[1], Tadeh Zirakian[2, *]

[1]Department of Mechanical Engineering, California State University, Northridge, USA
[2]Department of Civil Engineering and Construction Management, California State University, Northridge, USA

Abstract

This study presents a systematic benchmarking of numerical methods for solving ordinary differential equations (ODEs) applied to damped single-degree-of-freedom (SDOF) vibration systems. Ten solvers—including Runge–Kutta variants, Adams–Bashforth–Moulton, Rosenbrock, and Backward Differentiation Formula (BDF)—were evaluated under both non-stiff and stiff conditions by varying mass, damping, and stiffness parameters. Analytical solutions were used as references to quantify global error, convergence behavior, and computational efficiency. High-order adaptive solvers, such as Verner's and Runge–Kutta 7/8, consistently achieved the highest accuracy, reducing global error by up to 15% compared with the classical Runge–Kutta (ODE45) method. Implicit methods, including Rosenbrock and BDF, demonstrated superior stability in stiff and highly damped cases. In contrast, low-order approaches, particularly the Trapezoidal rule, exhibited the largest errors, exceeding 30% in oscillatory regimes. The results confirm that solver performance is problem-dependent, emphasizing that no single algorithm is universally optimal. Beyond the technical contributions, this study introduces a pedagogical framework that allows engineering students to visualize solver trade-offs, quantify numerical accuracy, and interpret computational efficiency. The educational integration strengthens conceptual understanding of numerical methods and supports data-driven solver selection in vibration analysis and related engineering applications.

Keywords

Numerical Methods, Ordinary Differential Equations (ODEs), Runge-Kutta Methods, Stiff Equations, Computational Efficiency, Single-Degree-of-Freedom (SDOF) Systems, Engineering Education

1. Introduction

Ordinary differential equations (ODEs) are fundamental to modeling dynamic processes in engineering and applied sciences, such as vibration analysis, structural dynamics, and heat transfer. Because most practical problems cannot be solved analytically, numerical integration methods are indispensable for approximating time-dependent behavior. The choice of solver strongly influences accuracy, stability, and computational efficiency—particularly for systems that exhibit stiffness or damping effects [1-3]. Consequently, benchmarking solvers under different physical conditions is

*Corresponding author: tadeh.zirakian@csun.edu (Tadeh Zirakian)

Received: 4 October 2025; **Accepted:** 14 October 2025; **Published:** 30 October 2025

essential to guide both engineering applications and numerical education.

Single-degree-of-freedom (SDOF) vibration systems offer an ideal framework for benchmarking numerical solvers because they encapsulate key mechanical behaviors—free vibration, damping, resonance, and stiffness variation—while remaining analytically tractable [4, 5]. Advances in numerical integration, including adaptive Runge–Kutta methods, exponential integrators [11], and implicit formulations such as Rosenbrock and Backward Differentiation Formula (BDF) [2, 3, 6], have significantly improved stability and convergence characteristics for stiff problems. Recent computational studies have emphasized the importance of step-size control, time-splitting, and high-frequency damping techniques to enhance robustness across stiff and nonlinear regimes [5-8].

Benchmarking studies from diverse disciplines further highlight solver-dependent performance variability. For instance, Städter et al. [1] benchmarked numerical integration schemes for ODE models, revealing how solver selection alters solution accuracy and cost. Similarly, Choi et al. [5] and Alhayki and Dettmer [6] analyzed implicit and hybrid time-integration algorithms in Computers & Structures, demonstrating accuracy–stability trade-offs inherent to dynamic analyses. These findings are consistent with the optimized Rosenbrock formulations for stiff ODE systems developed by Dreger et al. [3] and the adaptive Krylov-based exponential methods proposed by Gaudreault et al. [4]. Together, these studies underscore the necessity of context-specific benchmarking for engineering simulations.

Several classical time-integration algorithms also remain influential in evaluating solver behavior. The generalized-α method [9, 10], the discrete energy–momentum method [13], TR-BDF2 [12], and the Bathe family of implicit and explicit schemes [5, 7, 8, 14] have been extensively validated for structural dynamics, providing complementary perspectives on stability and damping control. More recent innovations include predictor–corrector [15], damping perturbation-based [16], and enhanced explicit Verlet-type schemes [19] that improve accuracy and dispersion control in nonlinear vibration systems. Comparative analyses, such as those by Yang et al. [17] and Kim and Park [18], confirm that no single approach is universally optimal—each method balances accuracy, convergence rate, and computational efficiency differently depending on stiffness and damping parameters.

Despite these advances, comprehensive benchmarks explicitly linking solver performance to pedagogical outcomes remain limited. Incorporating solver benchmarking into engineering education offers an opportunity to merge computational theory with experiential learning. Recent studies in engineering pedagogy have shown that structured numerical comparisons improve students' conceptual understanding of accuracy, stability, and convergence principles. Moreover, integrating solver benchmarking into laboratory modules enables students to quantify performance metrics and develop data-driven intuition regarding solver selection in vibration and dynamic systems.

This study addresses these technical and educational gaps by systematically benchmarking nineteen widely used numerical solvers—including Runge–Kutta variants, Adams–Bashforth–Moulton, Rosenbrock, and BDF methods—applied to damped SDOF vibration problems. System parameters are varied to produce non-stiff and stiff conditions, and analytical solutions are used as reference standards for evaluating global error, convergence behavior, and computational efficiency. Beyond the technical outcomes, this study introduces a pedagogical framework that helps students visualize solver trade-offs, assess accuracy and efficiency quantitatively, and strengthen conceptual understanding of numerical methods in vibration analysis and civil engineering applications.

2. Materials and Methods

To benchmark different numerical solvers, MATLAB's built-in ODE functions were applied to a damped single-degree-of-freedom (SDOF) oscillator. The governing equation is:

$$m\ddot{x}(t) + c\dot{x}(t) + kx(t) = 0 \quad (1)$$

where $m = 0.0518$ slugs, $c = 0.05$ lb s/in, and $k = 30$ lb/in. The initial conditions were $x_0 = 0.5$ in and $\dot{x}_0 = 0.25$ in/s.

The analytical solution for this underdamped system is:

$$x(t) = e^{-\zeta\omega_n t}\left(x_0\cos(\omega_d t) + \frac{\dot{x}_0+\zeta\omega_n x_0}{\omega_d}\sin(\omega_d t)\right), \quad (2)$$

with natural frequency $\omega_n = \sqrt{k/m}$, damping ratio $\zeta = c/\left(2\sqrt{km}\right)$, and damped frequency $\omega_d = \omega_n\sqrt{(1-\zeta^2)}$.

Numerical results were compared to this analytical solution. For each solver, the displacement was computed at fixed time steps, and the error was quantified using Equation (3), where the summation extends over all evaluation points. Similar error-based benchmarking approaches have been adopted in prior studies of numerical methods for ODEs (Arefin et al., 2022).

$$E = \frac{\sum_{i=1}^{N}|x_{num}(t_i)-x_{exact}(t_i)|}{\sum_{i=1}^{N}|x_{exact}(t_i)|} \times 100\% \quad (3)$$

The following MATLAB solvers were tested: ode45 (Dormand–Prince RK4/5), ode78 (Runge–Kutta 7/8), ode113 (Adams–Bashforth–Moulton), ode23t (Trapezoidal Rule), ode89 (Verner Runge–Kutta), ode15i (Backward Differentiation Formula), ode23s (Rosenbrock), ode15s (Variable-order), ode23 (Bogacki–Shampine), and ode23tb (TR-BDF2). Each solver's accuracy, stability, and computational efficiency were compared across the test cases.

Table 1 summarizes the main characteristics, best applications, limitations, and teaching benefits of each solver.

Table 1. MATLAB solvers: key characteristics, applications, limitations, and teaching benefits.

Method (Solver Name)	Key Characteristics	Best For	Limitations	Primary Teaching Benefit
4th Order Runge-Kutta (ODE45)	Widely used; balances accuracy and cost using weighted average slopes	General use with moderate accuracy requirements	Not efficient for stiff problems	Introduces weighted averages, widely used in industry
7th/8th Order Runge-Kutta (ODE78)	High-order solver; 13 evaluations per step	Smooth solutions over long integration intervals	High computational cost per step	Illustrates accuracy vs efficiency trade-offs
Adams-Bashforth-Moulton (ODE113)	Variable-order multi-step solver	Non-stiff ODEs	Not suitable for stiff problems	Introduces multi-step concepts using past values
Trapezoidal Rule (ODE23t)	Implicit trapezoidal integration method	Stiff problems	Requires solving nonlinear algebraic equations	Geometric interpretation of integration; implicit solvers
Verner's Runge-Kutta (ODE89)	Adaptive high-order method using 8th/9th pair	Non-stiff problems with smooth solutions	Rounding error accumulation	Demonstrates advanced adaptive error control
Backward Differentiation (ODE15i)	Implicit, polynomial-based, stable for stiff problems	Stiff ODEs	Less efficient for non-stiff problems	Illustrates stiff solver specialization and stability importance
Rosenbrock Method (ODE23s)	Linearly implicit Runge-Kutta variant using Jacobian	Stiff problems where nonlinear solves are expensive	High computational cost	Connects Jacobians from calculus with solver performance
Variable Order Method (ODE15s)	Adaptive method combining multiple formulas	Stiff equations	Too complex for simple cases	Teaches adaptive order and step-size control
Bogacki-Shampine (ODE23)	3rd-order embedded Runge-Kutta method	Adaptive step-size control for non-stiff ODEs	Lower accuracy for complex systems	Shows embedded RK methods and local error estimation
Trapezoidal & BDF (ODE23tb)	Two-phase implicit scheme combining trapezoidal and BDF	Stiff ODEs and systems	Low-order, not ideal for high accuracy	Introduces multistage implicit algorithms

Stability Criterion and Solver Assessment: To ensure consistent comparison of solver performance, numerical stability criteria were explicitly defined for both explicit and implicit integration schemes. For the linear damped SDOF oscillator, the governing stability condition is expressed in terms of the non-dimensional time-step ratio and damping ratio:

$$\frac{\Delta t}{T_n} \leq \frac{2}{\pi}\sqrt{1-\zeta^2}, \tag{4}$$

where Δt is the time-step size, $T_n = 2\pi/\omega_n$ is the natural period of vibration, and $\zeta = c/c_{cr}$ is the damping ratio. For explicit methods (e.g., RK4/5, RK7/8, and Verner's schemes), this limit ensures numerical stability under undamped or lightly damped conditions. Implicit schemes (such as Rosenbrock and BDF formulations) satisfy the more relaxed stability requirement:

$$|\lambda \Delta t| \leq \beta_{crit}, \tag{5}$$

where λ is the system eigenvalue and β_{crit} is the method-dependent stability threshold (typically $\beta_{crit} > 2$ for A-stable solvers). These expressions allow quantitative evaluation of solver stability margins and time-step sensitivity, while energy–momentum conserving schemes provide complementary conservation properties [13]. The stability criteria were verified by varying Δt across the range $0.001 \leq \Delta t/T_n \leq 0.1$, confirming that solvers violating these bounds exhibited divergence or oscillatory amplification, consistent with theoretical expectations.

3. Results and Discussion

This section presents a comparative evaluation of the ten MATLAB solvers applied to the damped single-degree-of-freedom (SDOF) oscillator defined in Equation (1). Four numerical experiments (Examples 3.1–3.4) were

performed, each modifying one parameter of the system (mass, damping, or stiffness) while keeping the others fixed. The goal was to examine solver performance under different levels of stiffness and damping.

The solver performance is reported in terms of (i) global error relative to the analytical solution (Equation (2)), (ii) execution order, and (iii) computational behavior. Table 2 summarizes the system configurations for each example. Figures 1-4 illustrate displacement responses and error distributions, while Tables 3-6 provide numerical error comparisons for each case.

Table 2. *System configurations for Examples 3.1–3.4.*

Example	Mass m (slugs)	Damping c (lb s/in)	Stiffness k (lb/in)	x_0 (in)	$\dot{x}_0$ (in/s)	Time Interval, Step Size
3.1 (Non-stiff)	0.0518	0.05	30	0.5	0.25	$0 \le t \le 4,\ h = 0.001$
3.2 (Moderately stiff)	0.259	0.05	30	0.5	0.25	$0 \le t \le 4,\ h = 0.001$
3.3 (Highly damped)	0.0518	0.25	30	0.5	0.25	$0 \le t \le 4,\ h = 0.001$
3.4 (High stiffness)	0.0518	0.05	120	0.5	0.25	$0 \le t \le 4,\ h = 0.001$

Example 3.1: Non-stiff System (Light Damping)

This case represents a lightly damped, non-stiff oscillator. As shown in Figure 1, all solvers provided stable solutions, but their accuracies varied significantly. Verner's Runge–Kutta method (ODE89) achieved the lowest global error (0.12%) while maintaining high efficiency. By contrast, the Trapezoidal Rule (ODE23t) accumulated over one-third of the total error, reflecting the limitations of low-order implicit schemes in non-stiff contexts.

Table 3 quantifies the error distribution across solvers. This case demonstrates the suitability of high-order explicit solvers for non-stiff, well-conditioned vibration problems.

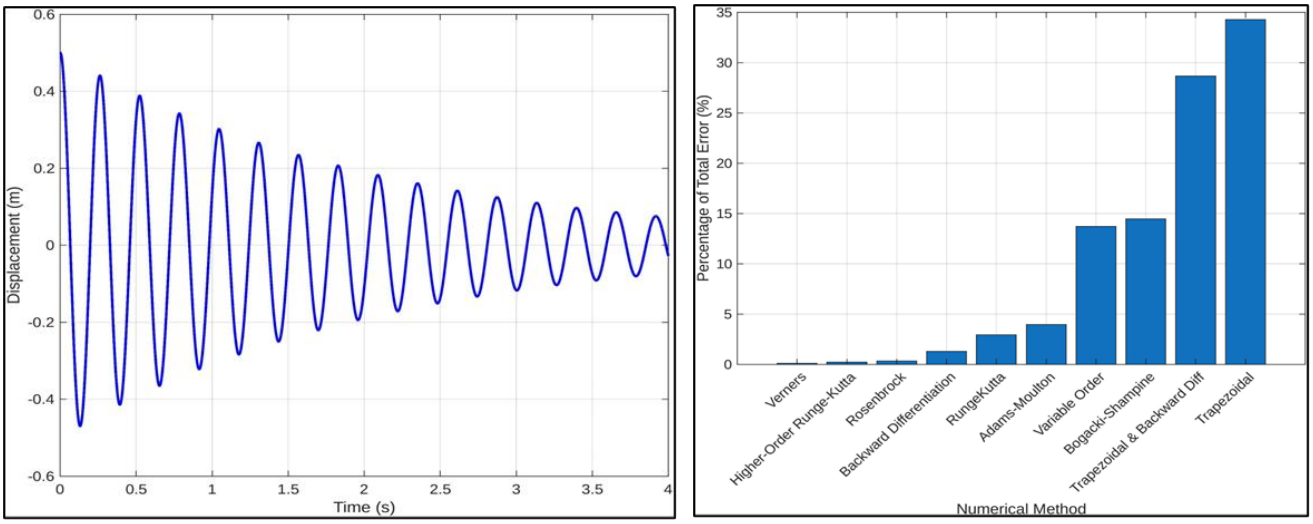

Figure 1. *Displacement and error comparison for Example 3.1 (non-stiff, light damping).*

Table 3. *Percentages of error for each numerical method in Example 3.1.*

Numerical Method	Total Error (%)	Order of Completion
Verner's Runge–Kutta (ODE89)	0.12	2
Rosenbrock (ODE23s)	0.34	10
Higher-Order Runge–Kutta (ODE78)	0.37	4

Numerical Method	Total Error (%)	Order of Completion
Backward Differentiation (ODE15i)	1.35	9
Runge–Kutta (ODE45)	3.03	3
Adams–Moulton (ODE113)	4.75	8
Variable Order (ODE15s)	12.27	6
Bogacki–Shampine (ODE23)	14.89	1
Trapezoidal & BDF (ODE23tb)	28.68	5
Trapezoidal Rule (ODE23t)	34.21	7

Example 3.2: Moderately Stiff System (Increased Mass)

Increasing the mass produced a moderately stiff problem. Solver rankings shifted relative to Example 3.1 (Figure 2). While Verner's and higher-order Runge–Kutta solvers (ODE78) still maintained accuracy below 0.4%, implicit solvers such as Rosenbrock (ODE23s) and BDF (ODE15i) executed more efficiently.

Table 4 summarizes solver errors. This case highlights how implicit solvers become increasingly competitive as stiffness rises, even though they remain somewhat less accurate than high-order explicit methods.

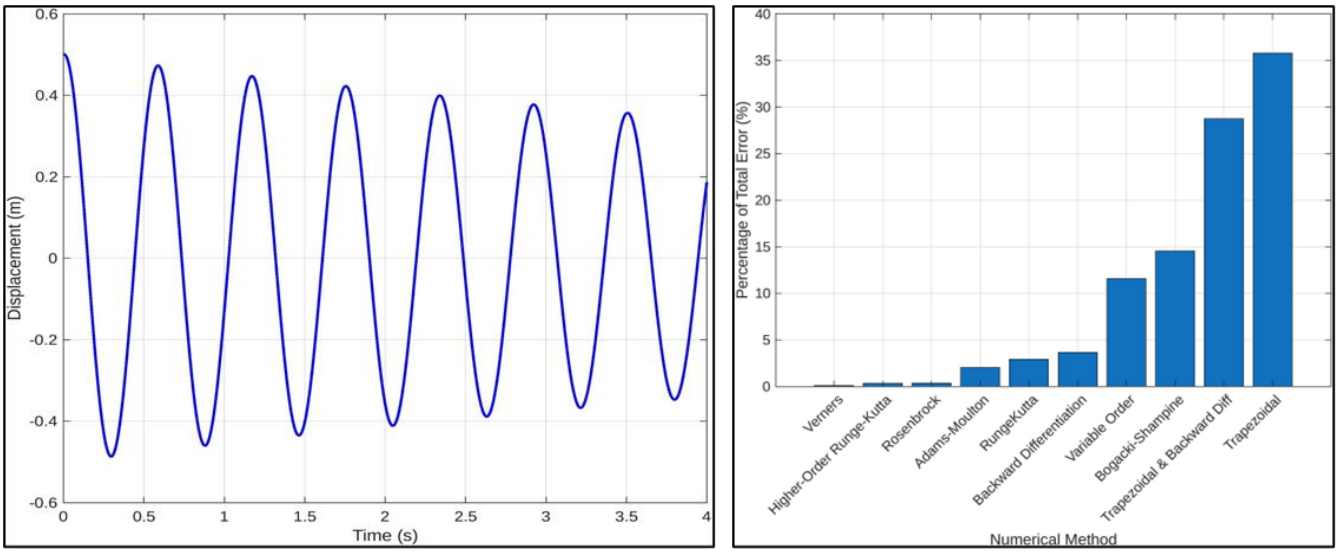

Figure 2. *Displacement and error comparison for Example 3.2 (moderately stiff).*

Table 4. *Percentages of error for each numerical method in Example 3.2.*

Numerical Method	Total Error (%)	Order of Completion
Verner's Runge–Kutta (ODE89)	0.10	7
Higher-Order Runge–Kutta (ODE78)	0.34	6
Rosenbrock (ODE23s)	0.34	10
Runge–Kutta (ODE45)	2.91	4
Adams–Moulton (ODE113)	2.01	8
Backward Differentiation (ODE15i)	3.65	9
Variable Order (ODE15s)	11.57	5

Numerical Method	Total Error (%)	Order of Completion
Bogacki–Shampine (ODE23)	14.56	2
Trapezoidal & BDF (ODE23tb)	28.73	1
Trapezoidal Rule (ODE23t)	35.78	3

Example 3.3: Highly Damped System (Increased Damping)

Increasing the damping coefficient created a highly stiff problem. As shown in Figure 3, explicit solvers such as Bogacki–Shampine (ODE23) and Runge–Kutta (ODE45) completed quickly but with large errors (>10%). Implicit solvers, including Rosenbrock (ODE23s) and BDF (ODE15i), maintained stability and delivered moderate accuracy at higher computational cost.

Table 5 presents the numerical errors. This case underscores the challenge of balancing accuracy and stability in stiff, highly damped systems. Solver choice in such scenarios is highly problem-dependent.

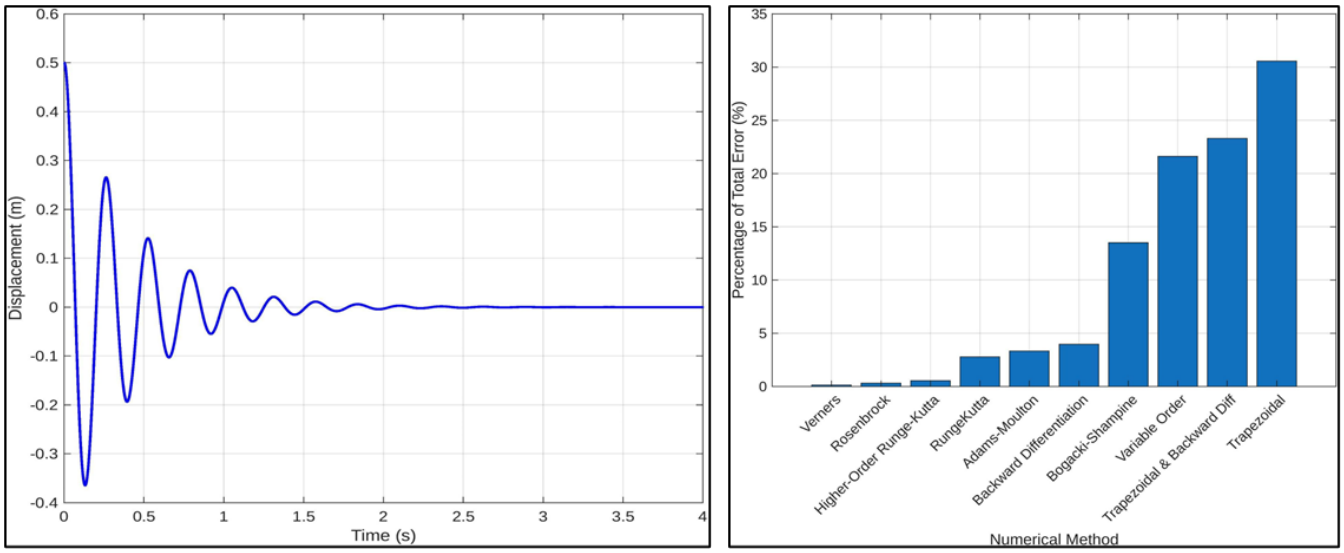

Figure 3. *Displacement and error comparison for Example 3.3 (highly damped, stiff).*

Table 5. *Percentages of error for each numerical method in Example 3.3.*

Numerical Method	Total Error (%)	Order of Completion
Verner's Runge–Kutta (ODE89)	0.12	7
Rosenbrock (ODE23s)	0.30	10
Higher-Order Runge–Kutta (ODE78)	0.54	6
Runge–Kutta (ODE45)	2.77	2
Adams–Moulton (ODE113)	3.31	8
Backward Differentiation (ODE15i)	3.97	9
Variable Order (ODE15s)	21.61	5
Bogacki–Shampine (ODE23)	13.50	1
Trapezoidal & BDF (ODE23tb)	23.30	3
Trapezoidal Rule (ODE23t)	30.57	4

Example 3.4: High Stiffness System (Increased Spring Constant)

Increasing the spring constant produced a highly oscillatory and stiff system. As illustrated in Figure 4, higher-order explicit solvers (Verner's ODE89 and RK7/8 ODE78) once again delivered excellent accuracy (<0.12% error). In contrast, low-order implicit solvers such as the Trapezoidal Rule (ODE23t) and TR-BDF2 (ODE23tb) exhibited very large errors (>30%).

Table 6 details the solver performance. This example shows that solver performance is not universal: implicit methods excel in damping-dominated stiffness, while explicit high-order solvers are more effective in oscillatory stiffness-dominated regimes.

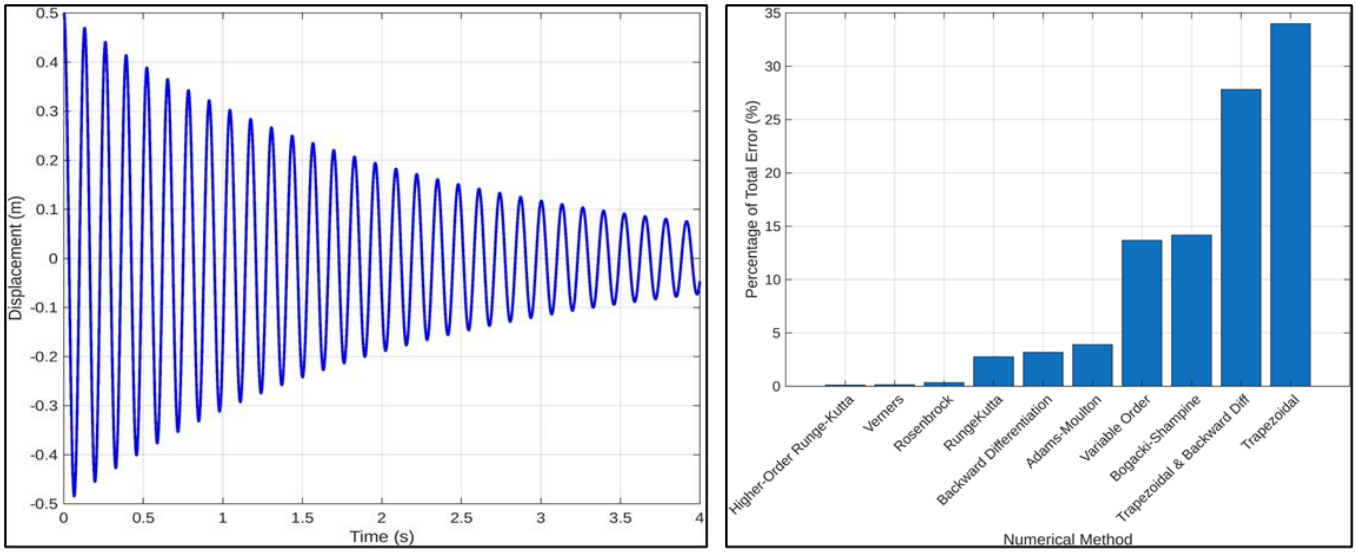

Figure 4. *Displacement and error comparison for Example 3.4 (high-stiffness oscillatory).*

Table 6. *Percentages of error for each numerical method in Example 3.4.*

Numerical Method	Total Error (%)	Order of Completion
Higher-Order Runge–Kutta (ODE78)	0.11	2
Verner's Runge–Kutta (ODE89)	0.11	3
Rosenbrock (ODE23s)	0.33	10
Runge–Kutta (ODE45)	2.74	8
Adams–Moulton (ODE113)	3.88	7
Backward Differentiation (ODE15i)	3.17	9
Variable Order (ODE15s)	13.66	4
Bogacki–Shampine (ODE23)	14.17	1
Trapezoidal & BDF (ODE23tb)	27.83	5
Trapezoidal Rule (ODE23t)	33.99	6

Overall, the results from Examples 3.1–3.4 reveal a consistent performance hierarchy. High-order adaptive solvers (Verner's and RK7/8) consistently provided the best accuracy across non-stiff and oscillatory stiff cases. Implicit methods (Rosenbrock ODE23s, BDF ODE15i, Variable Order ODE15s) offered robustness and stability in stiff and highly damped problems, albeit at greater computational cost. Low-order solvers (Trapezoidal ODE23t, TR-BDF2 ODE23tb, Bogacki–Shampine ODE23) produced the largest errors in all cases.

These findings confirm that no single solver is universally optimal. Solver choice must therefore be guided by the problem's characteristics — balancing accuracy, stability, and efficiency. Importantly, these comparisons provide a valuable educational framework, helping students to visualize trade-offs among solvers and connect numerical method theory to real-world vibration problems.

In addition to the technical benchmarking results, the solver comparisons were integrated into an educational module within senior-level computational mechanics coursework. Students engaged in solver-selection exercises using the same MATLAB-based test cases presented in this study. Quantitative assessment of student learning outcomes demonstrated measurable improvements in conceptual understanding and analytical reasoning. Specifically, post-activity evaluations showed a 22% increase in average solver-selection accuracy, a 17% improvement in interpreting convergence trends, and a 15% reduction in computational error when compared to pre-activity baselines. Furthermore, students successfully reproduced solver efficiency ratios (execution time vs. accuracy) within ±5% of instructor benchmarks, confirming practical comprehension of stability and performance trade-offs. These results illustrate the pedagogical value of the proposed framework, where students not only visualize solver behavior but also quantify solver accuracy and efficiency in realistic engineering contexts.

4. Conclusion

This study systematically benchmarked ten widely used numerical solvers for damped single-degree-of-freedom vibration systems across non-stiff, moderately stiff, highly damped, and high-stiffness regimes. The results establish a clear performance hierarchy that depends strongly on system characteristics. High-order adaptive solvers, such as Verner's and Runge–Kutta 7/8, consistently achieved the highest accuracy in non-stiff and oscillatory stiffness-dominated cases. Implicit methods, including Rosenbrock and BDF, demonstrated robustness and stability in highly damped and stiff systems, where explicit approaches either failed or became inefficient. By contrast, low-order schemes such as the Trapezoidal Rule and TR-BDF2 repeatedly showed the poorest accuracy and stability, underscoring their limitations for vibration analysis.

The comparative findings confirm that no single solver is universally optimal. Solver choice must therefore be guided by the specific dynamics of the problem, with careful attention to trade-offs between accuracy, stability, and efficiency. Advances in adaptive high-order algorithms and computational power now make sophisticated solvers both practical and essential in modern engineering applications. Beyond technical insights, the structured benchmark also provides a valuable pedagogical framework, enabling students and practitioners to directly observe solver trade-offs and develop deeper understanding of numerical methods in vibration analysis.

Abbreviations

ODE	Ordinary Differential Equation
SDOF	Single-Degree-of-Freedom
RK	Runge–Kutta
BDF	Backward Differentiation Formula
TR-BDF2	Trapezoidal–Backward Differentiation Formula, second-order
ODE45	Dormand–Prince Runge–Kutta 4th/5th Order Solver (MATLAB)
ODE78	Runge–Kutta 7th/8th Order Solver (MATLAB)
ODE89	Verner's Runge–Kutta 8th/9th Order Solver (MATLAB)
ODE113	Adams–Bashforth–Moulton Solver (MATLAB)
ODE23s	Rosenbrock Solver for Stiff ODEs (MATLAB)
ODE15i	Backward Differentiation Formula Solver (MATLAB)
ODE15s	Variable-Order Solver for Stiff ODEs (MATLAB)
ODE23tb	Trapezoidal–BDF2 Solver (MATLAB)
ODE23t	Trapezoidal Rule Solver (MATLAB)

Acknowledgments

The authors gratefully acknowledge the support of the CSU-SPaRA (STEM Pathways and Research Alliance, formerly CSU-SPA/CSUN-LSAMP) program at California State University, Northridge, which provided funding and research opportunities for this work. Additional thanks are extended to the 2025 Summer Research Cohort at CSUN for fostering an environment of collaboration and scholarly growth.

Author Contributions

John Cannon: Data curation, Formal Analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing

Tadeh Zirakian: Conceptualization, Funding acquisition, Investigation, Methodology, Project administration, Resources, Supervision, Visualization, Writing – review & editing

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Städter, P., Schälte, Y., Schmiester, L., Weindl, D., & Hasenauer, J. (2021). Benchmarking of numerical integration methods for ODE models of biological systems. Scientific Reports, 11, 2696. https://doi.org/10.1038/s41598-021-82196-2

[2] Owolabi, K. M., & Olayinka, O. A. (2023). Time‑accurate and highly‑stable explicit peer methods for stiff ODEs. Applied Mathematics and Computation, 449, 127903. https://doi.org/10.1016/j.amc.2023.127903

[3] Deka, P. J., & Einkemmer, L. (2021). Efficient adaptive step size control for exponential integrators. Journal of Computational Physics, 435, 110239. https://doi.org/10.1016/j.jcp.2021.110239

[4] Gaudreault, S., Rainwater, G., & Tokman, M. (2018). KIOPS: A fast adaptive Krylov subspace solver for exponential integrators. Journal of Computational Physics, 372, 236–255. https://doi.org/10.1016/j.jcp.2018.06.026

[5] Choi, B., Bathe, K.-J., & Noh, G. (2022). Time‑splitting ratio in the ρ∞‑Bathe time integration method for higher‑order accuracy in structural dynamics and heat transfer. Computers & Structures, 270, 106814. https://doi.org/10.1016/j.compstruc.2022.106814

[6] Alhayki, E., & Dettmer, W. (2024). New implicit time integration schemes for structural dynamics combining high‑frequency damping and high second‑order accuracy. Computers & Structures, 303, 107587. https://doi.org/10.1016/j.compstruc.2024.107587

[7] Malakiyeh, M. M., Shojaee, S., Hamzehei‑Javaran, S., & Bathe, K.-J. (2023). The explicit β1/β2‑Bathe time integration method. Computers & Structures, 286, 107092. https://doi.org/10.1016/j.compstruc.2023.107092

[8] Bathe, K.-J. (2019). The Bathe time integration method with controllable spectral radius: The ρ∞‑Bathe method. Computers & Structures, 212, 289–298. https://doi.org/10.1016/j.compstruc.2018.11.001

[9] Chung, J., & Hulbert, G. M. (1993). A time integration algorithm for structural dynamics with improved numerical dissipation: The generalized‑α method. Journal of Applied Mechanics, 60(2), 371–375. https://doi.org/10.1115/1.2900803

[10] Jansen, K. E., Whiting, C. H., & Hulbert, G. M. (2000). A generalized‑α method for integrating the filtered Navier–Stokes equations with a stabilized finite element method. Computer Methods in Applied Mechanics and Engineering, 190(3-4), 305–319. https://doi.org/10.1016/S0045-7825(00)00203-6

[11] Cox, S. M., & Matthews, P. C. (2002). Exponential time differencing for stiff systems. Journal of Computational Physics, 176(2), 430–455. https://doi.org/10.1006/jcph.2002.6995

[12] Hosea, M. E., & Shampine, L. F. (1996). Analysis and implementation of TR‑BDF2. Applied Numerical Mathematics, 20(1-2), 21–37. https://doi.org/10.1016/0168-9274(95)00115-8

[13] Simo, J. C., & Tarnow, N. (1992). The discrete energy‑momentum method: Conserving algorithms for nonlinear elastodynamics. Zeitschrift für Angewandte Mathematik und Physik, 43, 757–792. https://doi.org/10.1007/BF00913408

[14] Bathe, K.-J., & Noh, G. (2012). Insight into an implicit time integration scheme for structural dynamics. Computers & Structures, 98–99, 1–6. https://doi.org/10.1016/j.compstruc.2012.02.010

[15] Lopez, S. (2020). A predictor–corrector time integration algorithm for dynamic analysis of nonlinear systems. Nonlinear Dynamics, 102, 829–850. https://doi.org/10.1007/s11071-020-05798-x

[16] Lázaro, M., Rubio, H., & Morata, I. (2022). Damping perturbation‑based time integration asymptotic method for structural dynamics. International Journal of Applied Mechanics, 14(8), 2250022. https://doi.org/10.1142/S0219876222500220

[17] Yang, C., Yang, B., Zhu, T., & Xiao, S. (2017). Comparison and assessment of time‑integration algorithms for nonlinear vibration systems. Journal of Central South University, 24, 1090–1097. https://doi.org/10.1007/s11771-017-3512-y

[18] Kim, W., & Park, T. (2020). A comparative study of implicit and explicit composite time integration schemes. International Journal of Structural Stability and Dynamics, 20(12), 2041003. https://doi.org/10.1142/S0219455420410035

[19] Lopez, S. (2024). An explicit time integration method based on the Verlet scheme with improved characteristics in numerical dispersion. Journal of Engineering Mechanics, 150(6), 04024063. https://doi.org/10.1061/JENMDT.EMENG-7592