IMPROVED TIME INTEGRATION ALGORITHMS FOR THE ANALYSIS OF STRUCTURAL DYNAMICS

A Dissertation

by

WOORAM KIM

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	J. N. Reddy
Committee Members,	Ramesh Talreja
	Chii-Der Steve Suh
	Sevan Goenezen
Head of Department,	Andreas A. Polycarpou

August 2016

Major Subject: Mechanical Engineering

ABSTRACT

In this dissertation one family of second-order and two families of higher-order time integration algorithms are newly developed.

For the development of a new family of second-order time integration algorithms, the original equation of structural dynamics is rewritten as two first order differential equations and one algebraic equation. The these equations are called mixed formulations, because they include three different kinds of dependent variables (i.e., the displacement, velocity, and acceleration vectors). Equal linear (for the first sub-step) and quadratic (for the second sub-step) Lagrange type interpolation functions in time are used to approximate all three variables involved in the mixed formulations, then the time finite element method and the collocation method are applied to the velocity-displacement and velocity-acceleration relations of the mixed formulations to obtain one- and two-step time integration schemes, respectively. Newly developed one- and two-step time integration schemes are combined as one complete algorithm to achieve enhanced computational features. Two collocation parameters, which are included in the complete algorithm, are optimized and restated in terms of the spectral radius in the high frequency limit (also called the ultimate spectral radius) for the practical control of algorithmic dissipation. Both linear and non-linear numerical examples are analysed by using the new algorithm to demonstrate enhanced performance of it. The newly developed second-order algorithm can include the Baig and Bathe method and the non-dissipative case as special cases of its family.

For the development of the first family of higher-order time integration algo-

rithms, the displacement vector is approximated over the time interval by using the Hermite interpolation functions in time. The residual vector is defined by substituting the approximated displacement vector into the equation of structural dynamics. The modified weighted residual method is applied to the residual vector. The weight parameters are used to restate the integral forms of the weighted residual statements as algebraic forms, then these parameters are optimized by using the single-degree-of-freedom problem and its exact solution to achieve improved accuracy and unconditional stability. Numerical examples are used to verify performances of the new algorithms.

For the development of the second family of implicit higher-order time integration algorithms, the mixed formulations that include three time dependent variables (i.e., the displacement, velocity and acceleration vectors) are used. The equal degree Lagrange type interpolation functions in time are used to approximate the dependent variables involved in the mixed formulations, and the time finite element method and the modified weighted residual method are applied to the velocity-displacement and velocity-acceleration relations of the mixed formulations. Weight parameters are used and optimized to achieve preferable attributes of time integration algorithms. Specially considered numerical examples are used to discuss some fundamental limitations of well-known second-order accurate algorithms and to demonstrate advantages of using newly developed higher-order algorithms.

DEDICATION

To

My respectable grandparents, Dal-Soo Yoon, and Seung-Im Paek

My devoted mother, Ra-Kyoung Yoon

My dependable father-in-law, Ki-Il Son

My beloved wife, Donghee Son

My beautiful sons, Dan and Chan

and

My exemplary uncles, Sang-Kook Yoon, Sang-Eok Yoon, Sang-Man Yoon, and Sang-Don Yoon, who have guided me through the darkness to the brightness.

ACKNOWLEDGEMENTS

I wish to express my sincere thanks and heartfelt gratitude to my advisor Dr. J. N. Reddy for his support during the course of my M.S. and Ph.D. studies at Texas A&M University. His dedication to teaching and devotion to students have always been an examplary character of a true educator, and his passion for a pursuit of the truths of knowledge always inspired and stimulated me to proceed toward the final goal of my study.

Special appreciation is extended to my committee members, Professor Ramesh Talreja, Professor Steve Suh, Professor Sevan Goenezen, for their valuable comments and suggestions on this dissertation and their support throughout the course of this research.

I am particularly grateful for the financial supports which have been provided to me during my Ph.D. studies by the Oscar S. Wyatt Endowed Chair and the Aruna and J. N. Reddy Graduate Fellowship in Applied and Computational Mechanics at Texas A&M University. This study has also been greatly supported by the Republic of Korea Army.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

One common strategy of analysing partial differential equations (PDEs) associated with structural dynamics is to discretize the spatial domain of the PDEs first based on separation of variables [1, 2, 3]. After spatially discretizing the PDEs by employing proper numerical methods, a set of ordinary differential equations (ODEs) in time is obtained. This set of ODEs is called the semi-discrete equation of motion [1, 4, 5, 6] or the equation of linear structural dynamics [7, 8]. If the PDEs are linear, the semi-discrete system can be written as

$$\mathbf{M}\ddot{\mathbf{u}}(t) + \mathbf{C}\dot{\mathbf{u}}(t) + \mathbf{K}\mathbf{u}(t) = \mathbf{f}(t) \tag{1.1}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{C}$ is the viscous damping matrix, $\mathbf{K}$ is the stiffness matrices, $\mathbf{f}(t)$ is the vector of applied forces, $\mathbf{u}(t)$ is the displacement vector, $\dot{\mathbf{u}}(t)$ is the velocity vector, and $\ddot{\mathbf{u}}(t)$ is the acceleration vector. A solution of the initial value problem described by Eq. (1.1) satisfies the following initial conditions

$$\mathbf{u}(0) = \mathbf{u}_0 \tag{1.2a}$$

$$\dot{\mathbf{u}}(0) = \mathbf{v}_0 \tag{1.2b}$$

where $\mathbf{u}_0$ and $\mathbf{v}_0$ are the initial displacement and velocity vectors, respectively.

There are two general strategies which can be used to analyze Eq. (1.1). The exact solution can be obtained by using the modal decomposition method. Otherwise, the numerical solution of Eqs. (1.1)-(1.2) can be found by using proper step-by-step direct time integration algorithms.

The modal decomposition method is very useful for simple linear systems, but it is not suitable for complicated nonlinear systems. In addition to these limitations, special knowledge of the solution method is required in the modal decomposition method.

On the other hand, direct time integration algorithms are more broadly used than the modal decomposition method, because they can be systematically applied to general second-order initial value problems without special knowledge of solution method. In direct integration algorithms, numerical solutions of Eqs. (1.1) and (1.2) can be easily found by just inputting the given data (i.e., $\mathbf{M}$, $\mathbf{C}$, $\mathbf{K}$, $\mathbf{f}(t)$, $\mathbf{u}_0$, and $\mathbf{v}_0$) of the problems. In addition, majority of the existing direct time integration algorithms can be applied to nonlinear analyses without any modifications. In fact, many direct time integration algorithms were developed mainly for nonlinear analyses. Due to these advantages, the development of improved time integration algorithms has long been an interest of structural dynamics. Some applications of direct time integration algorithms to various types of challenging problems can be found in Refs [9, 10, 11, 12, 13].

Time integration algorithms can be categorized into explicit and implicit algorithms according to Ref. [1, 14, 15, 16, 17, 18, 19]. Explicit algorithms do not require factorization of the effective stiffness matrix to advance a time step, while implicit algorithms do. Naturally, less computational effort is required in explicit algorithms compared to implicit algorithms. However, explicit algorithms are only conditionally stable, thus, time steps should be small enough to satisfy the stability condition. In explicit algorithms, the critical sizes of time steps are inversely proportional to the highest frequency of the discrete system. If a chosen time step is larger than the critical time step of a chosen explicit algorithm, then the algorithm become unstable.

In many cases, it is very difficult to accurately discretize spatial domains of given

problems, and poor discretization of spatial domains often introduce artifacts called the spurious high frequency modes [20, 21, 22]. A popular remedy for this artifact is to eliminate the spurious high frequency responds from numerical solutions by utilizing the numerical damping of time integration algorithms. However, explicit algorithms may amplify the spurious high frequency responds, worsening the stability, rather than providing algorithmic damping in the high frequency range. As a matter of fact, time steps should be chosen as inversely proportional to the spurious high frequency in order to secure stability in explicit algorithms. This also means that time steps may become unnecessarily small depending on the value of high frequency, and the computational cost may rise up to unaffordable levels in some extreme cases.

However, implicit algorithms are often preferred to explicit algorithms because they can be designed as unconditionally stable ones. As a result, choices of time steps are not restricted by stability conditions in most of implicit algorithms. In addition to unconditional stabilities, implicit algorithms can be designed to possess numerical damping in the high frequency ranges. With proper sizes of time steps, unconditionally stable implicit algorithms designed to possess numerical damping in the high frequency regime can effectively filter out the spurious high frequency responds without additional filtering algorithms.

1.1 Second-Order Time Integration Algorithms

Some of second-order algorithms are still being broadly used for dynamic analyses of structural problems due to reasonable accuracy, affordable computational cost, and simple computer implementation. Here, we briefly review some implicit algorithms which have been proven to be effective ones through years of uses in linear and nonlinear structural dynamics.

After the introductions of the Houbolt method [9] and the Newmark method

[23], several improved time integration algorithms were developed. The Wilson θ-method [24, 25], the Park method [26], the collocation method [27], the modified Houbolt method [28], the ρ-method [29] were developed to achieve some of preferable attributes, such as unconditional stability, improved accuracy, and numerical dissipation in the high frequency range. Later, the generalized single-step method [30, 31] and the θ_1 method [32] were designed to included most of the previous algorithms as special cases of them, and higher-order accurate cases were also obtained by increasing the degree of the approximations. According to Ref. [33], however, higher-order algorithms obtained from the generalized single-step and θ_1 methods are only conditionally stable.

The Newmark method can be considered as the most significant second-order algorithm, because the truncated finite difference approximations used in the Newmark method have been used as bases of several improved second-order algorithms. For example, the HHT-α method [20], the WBZ-α method [34], and the generalized-α method [7] used exactly the same finite difference approximations used in the Newmark method. In the original Newmark method, the dynamic equilibrium equation was obtained by evaluating all force members (i.e., the inertia, viscous damping, and internal stress related forces) of the equation of structural dynamics at the end of the time interval. On the contrary to the Newmrk method, the improved algorithms listed above used modified dynamic equilibrium equations with additional parameters, which are called the alpha parameters. In the modified dynamic equilibrium equations, at least one of force members was allowed to vary linearly within the time interval, and the variations of the force members were adjusted through the alpha parameters. By optimizing the alpha parameters improved performances were obtained. Among the improved algorithms listed above, only the generalized-α method can provide the second-order accuracy for numerically dissipative cases, the

unconditional stability, and the controllable algorithmic dissipation.

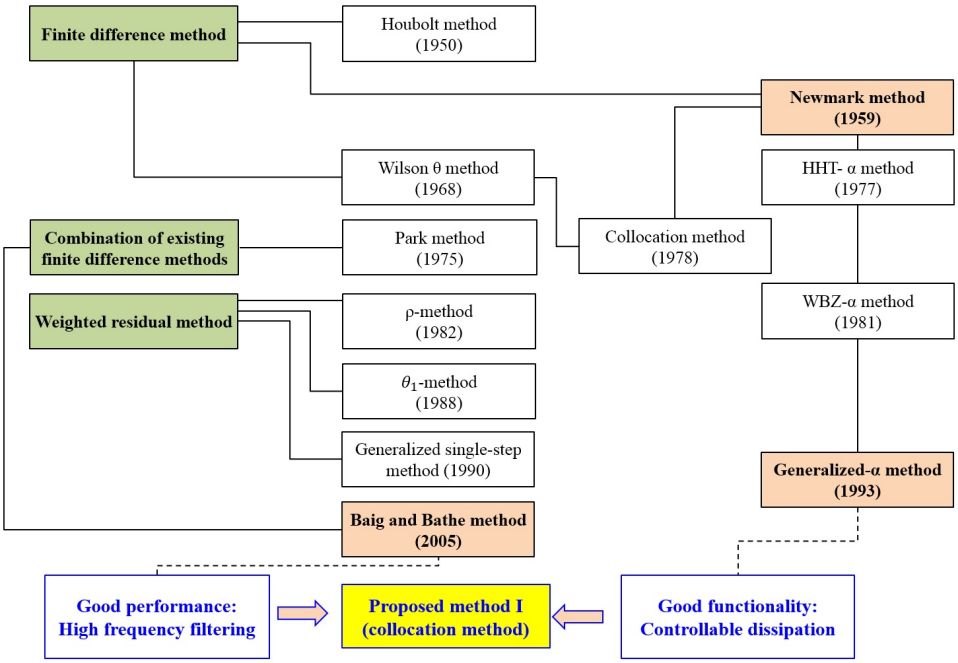

Figure 1.1: Second-order algorithms

Recently, the Baig and Bathe method [35] was introduced. The Baig and Bathe method possess very strong algorithmic damping in the high frequency regimes. This special attribute is also observed in the Hobolt and Park methods, and sometimes called the asymptotic annihilation property [36, 7]. If an algorithm has the asymptotic annihilation property, it can filter out any effects coming from the high frequency regime within one time step. In the Baig and Bathe method, the asymptotic annihilating property was enhanced by combining two well-known method.

For a better understanding of second-order algorithms, the Newmark method, the generalised-α method, and the Baig and Bathe method are reviewed briefly.

1.1.1 Newmark Method

The Newmark method [23] is one of the most broadly used single step second-order time integration algorithms. After the introduction of the Newmark method in 1959, its truncated finite difference approximations were used in the improved algorithms [7, 28, 34, 37, 38]. In the Newmark method, the dynamic equilibrium equation is satisfied at the end of the time interval (i.e., at $t = t_{s+1}$) as follows:

$$\mathbf{M}\ddot{\mathbf{u}}_{s+1} + \mathbf{C}\dot{\mathbf{u}}_{s+1} + \mathbf{K}\mathbf{u}_{s+1} = \mathbf{f}(t_{s+1}) \tag{1.3}$$

Here, $\mathbf{u}_{s+1}$ is the displacement vector at $t = t_{s+1}$. In the Newmark method, the displacement and velocity vectors at $t = t_s + \Delta t$ are obtained from the Taylor's series expressions of

$$\mathbf{u}_{s+1} = \mathbf{u}_s + \Delta t\ \dot{\mathbf{u}}_s + \frac{1}{2}\Delta t^2\ \ddot{\mathbf{u}}_s + \frac{1}{6}\Delta t^3\ \dddot{\mathbf{u}}_s + \cdots \tag{1.4a}$$

$$\dot{\mathbf{u}}_{s+1} = \dot{\mathbf{u}}_s + \Delta t\ \ddot{\mathbf{u}}_s + \frac{1}{2}\Delta t^2\ \dddot{\mathbf{u}}_s + \cdots \tag{1.4b}$$

Eqs. (1.4a) and (1.4b) are truncated as

$$\mathbf{u}_{s+1} = \mathbf{u}_s + \Delta t\ \dot{\mathbf{u}}_s + \frac{1}{2}\Delta t^2\ \ddot{\mathbf{u}}_s + \beta\Delta t^3\ \dddot{\mathbf{u}} \tag{1.5a}$$

$$\dot{\mathbf{u}}_{s+1} = \dot{\mathbf{u}}_s + \Delta t\ \ddot{\mathbf{u}}_s + \gamma\Delta t^2\ \dddot{\mathbf{u}} \tag{1.5b}$$

where β and γ are the two truncation parameters, and $\mathbf{u}_{s+1}$ is the displacement vector at $t = t_s$, $\Delta t = t_{s+1} - t_s$ being the size of the time step.

In the Newmark method, $\dddot{\mathbf{u}}$ is assumed to be

$$\dddot{\mathbf{u}} = \frac{\ddot{\mathbf{u}}_{s+1} - \ddot{\mathbf{u}}_s}{\Delta t} \tag{1.6}$$

By using Eqs. (1.5a) and (1.5b), two of $\ddot{\mathbf{u}}_{s+1}$, $\dot{\mathbf{u}}_{s+1}$, and $\mathbf{u}_{s+1}$ can be eliminated from Eq. (1.3), and the remaining one unknown vector can be solved. After finding one of three unknown vectors, remaining two unknown vectors are updated by using Eqs. (1.5a) and (1.5b).

The Newmark method can also include some well-known finite difference methods as special cases. In the Newmark method, the choice of $\beta = \frac{1}{4}$ and $\gamma = \frac{1}{2}$ gives the constant average acceleration method (also known as the trapezoidal rule), which is second-order accurate and unconditionally stable, and the choice of $\beta = \frac{1}{6}$ and $\gamma = \frac{1}{2}$ gives the linear acceleration method, which is third-order accurate and conditionally stable.

1.1.2 Genralized-α Method

The generalized-α method [17, 7, 39] is one of the latest alpha type modified methods. The generalized-α method was developed based on the modified dynamic equilibrium equation and the Newmark approximations given in Eqs. (1.5a) and (1.5b). In this method, the dynamic equilibrium equation is satisfied at some intermediate time points within the time interval Δt. The modified dynamic equilibrium equation of the generalized-α method is given by

$$\mathbf{M}\ddot{\mathbf{u}}_{s+1-\alpha_m} + \mathbf{C}\dot{\mathbf{u}}_{s+1-\alpha_f} + \mathbf{K}\mathbf{u}_{s+1-\alpha_f} = \mathbf{f}(t_{s+1-\alpha_f}) \tag{1.7}$$

where $\ddot{\mathbf{u}}_{s+1-\alpha_m}$, $\dot{\mathbf{u}}_{s+1-\alpha_f}$, $\mathbf{u}_{s+1-\alpha_f}$ and $t_{s+1-\alpha_f}$ are defined as

$$\mathbf{u}_{s+1-\alpha_f} = (1-\alpha_f)\ \mathbf{u}_{s+1} + \alpha_f\ \mathbf{u}_s \tag{1.8a}$$

$$\dot{\mathbf{u}}_{s+1-\alpha_f} = (1-\alpha_f)\ \dot{\mathbf{u}}_{s+1} + \alpha_f\ \dot{\mathbf{u}}_s \tag{1.8b}$$

$$\ddot{\mathbf{u}}_{s+1-\alpha_m} = (1-\alpha_m)\ \ddot{\mathbf{u}}_{s+1} + \alpha_m\ \ddot{\mathbf{u}}_s \tag{1.8c}$$

$$t_{s+1-\alpha_f} = (1-\alpha_f)\ t_{s+1} + \alpha_f\ t_s \tag{1.8d}$$

After substituting Eqs. (1.8a)-(1.8d) into Eq. (1.7), Eq. (1.7) can be restated in terms of $\ddot{\mathbf{u}}_{s+1}$, $\dot{\mathbf{u}}_{s+1}$. Then, two of $\ddot{\mathbf{u}}_{s+1}$, $\dot{\mathbf{u}}_{s+1}$, and $\mathbf{u}_{s+1}$ can be eliminated from Eq. (1.7) by using the Newmark approximations given in Eqs. (1.5a) and (1.5b). The remaining unknown vector can be found by solving Eq. (1.7), and the others can be updated by using Eqs. (1.5a) and (1.5b). This procedure is very similar to the procedure explained in the Newmark method.

In the generalized-α method, γ, β, α_f and α_m have been optimized to achieve second-order accuracy, unconditional stability, and minimized low-frequency dissipation. In this method, all four parameters have been optimized as

$$\gamma = \frac{1}{2} - \alpha_m + \alpha_f, \quad \beta = \frac{1}{4}\left(1 - \alpha_m + \alpha_f\right)^2, \quad \alpha_m = \frac{2\mu - 1}{\mu + 1}, \quad \alpha_f = \frac{\mu}{\mu + 1} \tag{1.9}$$

where μ is a user specified algorithmic dissipation control parameter which can range from 0 to 1. If $\mu = 1$, this algorithm becomes the non-dissipative algorithm (the trapezoidal rule), whereas $\mu = 0$ makes the algorithm as the asymptotic annihilation case, which gives the maximum algorithmic dissipation within its family. The controllable algorithmic dissipation and the second-order accuracy for dissipative cases have been considered as the main advantages of the generalized-α method.

1.1.3 Baig and Bathe Method

Two existing well-known methods have been combined in the Baig and Bathe method [35, 40, 41, 8]. In the Baig and Bathe method, the complete time interval Δt is subdivided into two sub-steps. Then the constant average acceleration method (the trapezoidal rule) and the 3-point Euler backward method are used for the first and second sub-steps, respectively. For the first sub-step ($t_s \leq t \leq t_{s+1/2}$), the dynamic equilibrium equation at $t = t_s + \frac{1}{2}\Delta t$ is given by

$$\mathbf{M}\ddot{\mathbf{u}}_{s+\frac{1}{2}} + \mathbf{C}\dot{\mathbf{u}}_{s+\frac{1}{2}} + \mathbf{K}\mathbf{u}_{s+\frac{1}{2}} = \mathbf{f}(t_{s+\frac{1}{2}}) \tag{1.10}$$

where $\mathbf{u}_{s+\frac{1}{2}}$ is the displacement vector at $t = t_s + \frac{1}{2}\Delta t$. Then $\dot{\mathbf{u}}_{s+\frac{1}{2}}$ and $\mathbf{u}_{s+\frac{1}{2}}$ are approximated as

$$\dot{\mathbf{u}}_{s+\frac{1}{2}} = \dot{\mathbf{u}}_s + \frac{\Delta t}{4}\left(\ddot{\mathbf{u}}_s + \ddot{\mathbf{u}}_{s+\frac{1}{2}}\right) \tag{1.11a}$$

$$\mathbf{u}_{s+\frac{1}{2}} = \mathbf{u}_s + \frac{\Delta t}{4}\left(\dot{\mathbf{u}}_s + \dot{\mathbf{u}}_{s+\frac{1}{2}}\right) \tag{1.11b}$$

For the second sub-step ($t_s \leq t \leq t_{s+1}$), the dynamic equilibrium equation at $t = t_s + \Delta t$ is given by

$$\mathbf{M}\ddot{\mathbf{u}}_{s+1} + \mathbf{C}\dot{\mathbf{u}}_{s+1} + \mathbf{K}\mathbf{u}_{s+1} = \mathbf{f}(t_{s+1}) \tag{1.12}$$

and, $\dot{\mathbf{u}}_{s+1}$ and $\ddot{\mathbf{u}}_{s+\frac{1}{2}}$ are approximated as

$$\dot{\mathbf{u}}_{s+1} = \frac{1}{\Delta t}\left(\mathbf{u}_s - 4\mathbf{u}_{s+\frac{1}{2}} + 3\mathbf{u}_{s+1}\right) \tag{1.13a}$$

$$\ddot{\mathbf{u}}_{s+1} = \frac{1}{\Delta t}\left(\dot{\mathbf{u}}_s - 4\dot{\mathbf{u}}_{s+\frac{1}{2}} + 3\dot{\mathbf{u}}_{s+1}\right) \tag{1.13b}$$

In the Baig and Bathe method, $\mathbf{u}_{s+\frac{1}{2}}$, $\dot{\mathbf{u}}_{s+\frac{1}{2}}$, and $\ddot{\mathbf{u}}_{s+\frac{1}{2}}$ are found by using Eqs. (1.10)-(1.11b). With $\mathbf{u}_{s+\frac{1}{2}}$, $\dot{\mathbf{u}}_{s+\frac{1}{2}}$, and $\ddot{\mathbf{u}}_{s+\frac{1}{2}}$ obtained in the first sub-step, $\mathbf{u}_{s+1}$, $\dot{\mathbf{u}}_{s+1}$, and $\ddot{\mathbf{u}}_{s+1}$ are found by solving Eqs. (1.12)-(1.13b). This method does not have any adjustable parameters, and performs only as the asymptotic annihilation case [36] similar to the Houbolt and the Park methods.

1.1.4 Some Limitations of Second-Order Algorithms

As mentioned, only the generalized-α method can provide (a) second-order accuracy, (b) unconditional stability, and (c) controllable algorithmic dissipation. In additions to these attributes, the generalized-α method is the only method which can retain the second-order accuracy for numerically dissipative cases. However, one shortcoming of the generalized-α method is that this method may introduce excessive numerical damping into the important low frequency mode as well as the spurious high frequency mode, if highly dissipative case is used. In other word, if the generalized-α method is used to eliminate the spurious high frequency mode from numerical solutions, the important low frequency mode in numerical solutions can also be filtered out.

If filtering of the high frequency modes in numerical solutions is the main purpose of an transient analysis, the Baig and Bathe method may be able to provide more accurate solutions than the generalized-α method, because the Baig and Bathe method can eliminate the spurious high frequency effect by introducing the maximum numerical damping into the high frequency range with the minimum damping into the important low frequency range. However, the Baig and Bathe method cannot adjust the level of the numerical damping, and it can perform as the asymptotic annihilation case only. The high frequency filtering capability is very useful in many cases of practical analyses, but there may be some situations where conservation of

the total energy of dynamic systems is more important. To handle different demands of various types of analyses more flexibly within a single computer code, a chosen time integration algorithm is required to possess the algorithmic dissipation capability. For this reason, the dissipation control capability has been considered to be a preferable attribute of time integration algorithms in many literatures [42, 43, 44]. In this viewpoint, the Baig and Bathe method is not a practical algorithm. However, the absence of undetermined parameters in the Baig and Bathe method has been considered an advantage of the method by the authors of Ref. [8].

In addition to the excessive numerical damping in low frequency mode in highly dissipative cases of the generalized-α method, applying the method to nonlinear analyses may require some additional modifications of nonlinearities included in the equation of structural dynamics, because this method uses the modified dynamic equilibrium equation as discussed. In the generalized-α method, a proper linearization of the internal force vector should be conducted to use the Newton-Raphson iterative method for the nonlinear equation solving as presented in Refs. [45, 46, 47]. On the other had, the Newmrk method and the Baig and Bathe method do not require any linearizations of the internal force vector to use the Newton-Raphson iterative method.

The shortcomings of two well-known algorithms can be overcome through a new algorithm. An improved algorithm can be developed by combining several well-established numerical techniques. In fact, the development of an improved second-order algorithm is still very important, because reasonably good solutions and affordable computational effort can be achieved with second-order algorithms in general analyses. However, there exist some analyses of extreme situations also, and second-order algorithms may not be able to handle them properly. To obtained acceptable predictions in some extreme transient analyses, second-order algorithms may use

very small sizes of time steps, but this may be accompany by a hugely increased computational cost.

Here, some fundamental limitations of second-order algorithms are discussed. First, second-order algorithms are not suitable for long-term analyses, because a second-order algorithm introduces a considerably large amount of error into the numerical solution in each time step. Since a step-by-step time integration algorithm uses the solution of the previous time step as the known data of the current equation solving, the solution of the current will be affected by the error of the previous solution as well as the algorithm. In a long-term analysis, this type of accumulation of errors can contaminate numerical solutions seriously even with a very small size of time step. In general, the quality of numerical solutions can be improved up to a certain level by reducing sizes of time steps. But in a long-term analysis, very small time steps may increase the computational cost up to an unaffordable level, if a given dynamic system is very large one.

Second, the range of admissible sizes of time steps is very narrow in dissipative second-order algorithms, if some situation requires the filtering of high-frequency modes. In many practical cases, the spatial discretization of given PDEs cannot be accurate enough to represent all exact frequency modes, poor representations of the spatial domain are often accompanied by the artifacts called the spurious high frequency modes. These spurious high frequency modes can affect the quality and stability (in some nonlinear cases) of numerical solutions. A simple and effective way of eliminating the spurious high frequency modes from numerical solutions is to utilize algorithmic dissipations of time integration algorithms. Some of the existing second-order algorithms, such as the Houbolt method, the Park method, and the Baig and Bathe method, can be used mainly for this purpose in practical analyses. They were designed to introduce the maximum numerical damping in the high frequency

limit, while minimizing the numerical damping in the important low frequency range. As a rule, the size of time step should be chosen properly from the range of $10\ T_H \leq \Delta t \leq \frac{1}{10}\ T_L$ in second-order algorithms, where T_H is the periods of the spurious high frequency mode, and T_L is the period of the important low frequency mode. If this condition is violated, dissipative second-order algorithms may give poor important low frequency solution or filter out spurious high frequency mode in a slow rate. Some of related discussions have been presented in Ref. [48].

Third, second-order algorithms cannot provide reliable nonlinear solutions for highly nonlinear problems as discussed in Ref. [11]. In highly nonlinear problems, numerical solutions obtained from second-order algorithms may contain noticeable amounts of period and amplitude errors, and these contaminated solutions are supposed to be used as the known properties in the next step to advance another step. For this reason, the next step solutions can be distorted more severely, if the current step solutions are inaccurate. Unlike the linear and moderately nonlinear problems, it is very difficult to determine a proper size of time step in highly nonlinear problems. Since time integration algorithms should be used repeatedly to obtain predictions at desired time point, analyses may become completely misleading one in highly nonlinear problems. Of course, majority of moderately nonlinear problems can be properly analyzed by using second order algorithms, and refining time steps can increase qualities of solutions in second-order algorithms. However, we do not have a proper measure which can be used as an indication for the time step refinement in nonlinear problems. In some highly nonlinear cases, numerical solutions of a given nonlinear problems may be a completely misleading ones, but user do not have an ability to check the validness of numerical solutions. In other words, using second order algorithms for the analyses of highly nonlinear problems is accompanied by the potential danger of getting a totally wrong prediction.

1.2 Higher-order Time Integration Algorithms

In general, the algorithm is called higher-order algorithm if the order of accuracy [49] of the time integration algorithm is higher than or equal to third. Over the past four decades, many higher-order algorithms have been developed based on various numerical methods to overcome the limitations of second-order algorithms. These numerical methods include the Newmark approximation [23] based sub-stepping methods [50, 51, 37, 52, 38, 53], the variational method [54, 55, 56, 57, 58], the weighted residual method [36, 59, 60, 61, 43], the collocation method [62, 63, 64, 65], and the differential quadrature method [66, 67, 44, 68]. Among the numerical method mentioned above, some can be used to develop higher-order algorithms of certain order of accuracies, while the others can be used to develop higher-order accurate algorithms of general order of accuracies.

Among several numerical methods which have been used to develop higher-order time integration algorithm families, only Fung's differential quadrature method [69, 44, 70, 68] and Fung's collocation method [71] can be used to develop time integration algorithms with general order of accuracy, controllable algorithmic dissipation, unconditional stability, and full extensibility to nonlinear cases. Both methods can provide equivalently accurate solutions for second-order initial value problems, and computer implementations of the final algorithms are very similar in both methods. However, the differential quadrature method can be considered more significant one, because it has simpler computational structure and better extensibility to various types of initial value problems.

The weighted residual method based algorithms proposed by Fung [72, 60] also have the controllable algorithmic dissipation, unconditional stability. But applying them to nonlinear analyses requires additional modification of algorithms or rear-

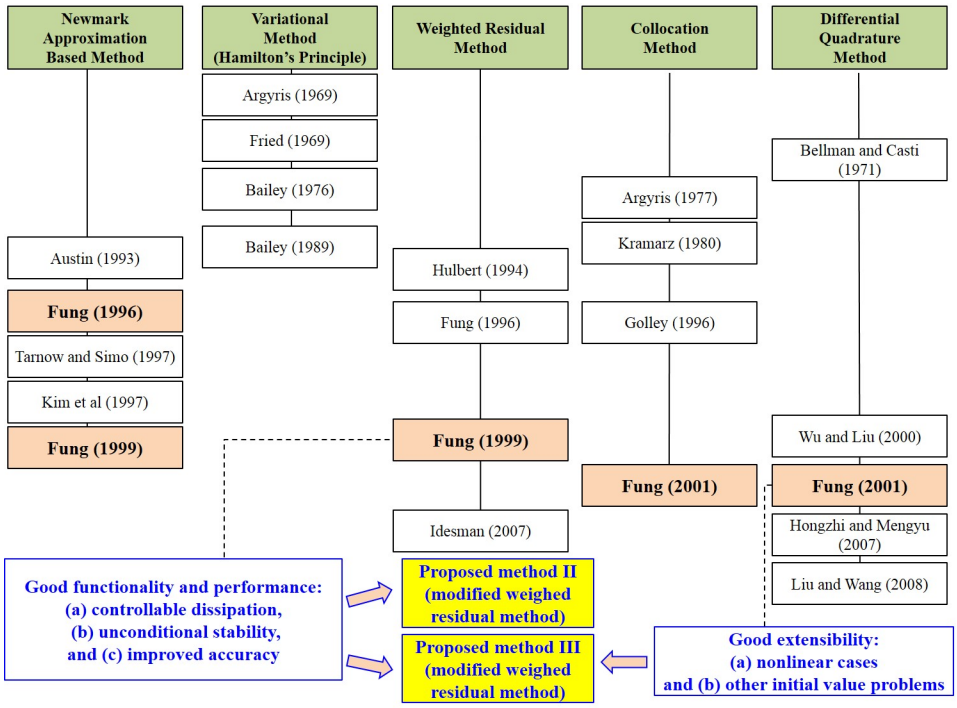

Figure 1.2: Higher-order algorithms

rangement of the governing equations into suitable forms for these algorithms. This is because the structural dynamics equation has been directly manipulated into the weighted integral forms in the minimization procedure of the residual. Naturally, the result equation obtained from the linear equation of structural dynamics cannot be directly used in nonlinear cases in the weighted residual method based algorithms. In fact, these weighted residual method based algorithms were not fully extended to nonlinear cases.

However, the differential quadrature method can be applied to general nonlinear problems, because discrete relations of a variable and its time derivatives are directly derived from known test functions. Once discrete time derivatives of the test function are stated in terms of the function values at the sampling points, they can be used

to discretize the equation of structural dynamics. Due to this simple characteristic of the differential quadrature method, both linear and nonlinear cases can be tackled without any modification of algorithms.

Here, the weighted residual and differential quadrature methods based higher-order time integration algorithms are briefly reviewed, because our new approaches are closely related with these two methods in conceptually and technically. More details of existing higher-order time integration algorithms are well summarized in Refs. [33], [73] and [74].

1.2.1 Weighted Residual Method Based Algorithms

Hulbert[36] rewrote the equation of structural dynamics as a set of two first-order equations, and the time discontinuous Galerkin method was applied to the rewritten equations. Two first-order equations used in time discontinuous Galerkin method are

$$\mathbf{M}\dot{\mathbf{v}}(t) + \mathbf{C}\mathbf{v}(t) + \mathbf{K}\mathbf{u}(t) = \mathbf{f}(t) \tag{1.14a}$$

$$\mathbf{v}(t) = \dot{\mathbf{u}}(t) \tag{1.14b}$$

$\mathbf{u}(t)$ and $\mathbf{v}(t)$ satisfy the initial conditions

$$\mathbf{u}(0) = \mathbf{u}_0 \tag{1.15a}$$

$$\mathbf{v}(0) = \mathbf{v}_0 \tag{1.15b}$$

Eqs. (1.14a) and (1.14b) can be called the mixed formulations [60, 75, 76, 77], because different types of variables are included in the formulations and approximated

independently. By applying the time discontinuous Galerkin Method to Eqs. (1.14a) and (1.14a), the unified set of single-step time integration algorithms was obtained. The algorithms developed by Hulbert did not possess dissipation control capability and performed as the asymptotic annihilation case only. Later Idesman also used Eqs. (1.14a) and (1.14a), but he employed the time continuous Galerkin method [43, 78]. Unlike Hulbert's algorithms, Idesman's algorithms were designed to possess dissipation control capability. However, the control of algorithmic dissipation was not intuitive in Idesman's algorithms. In both cases, nth-degree polynomials were used to approximate $\mathbf{u}(t)$ and $\mathbf{v}(t)$, and Eqs. (1.14a) and (1.14a) were used to define residual vectors in time. In Idesman's work [43], the displacement and the velocity vectors were approximated as

$$\tilde{\mathbf{u}}(t) = \mathbf{u}_0 + \mathbf{u}_1\, t + \mathbf{u}_2\, t^2 + \cdots + \mathbf{u}_n\, t^n \tag{1.16a}$$

$$\tilde{\mathbf{v}}(t) = \mathbf{v}_0 + \mathbf{v}_1\, t + \mathbf{v}_2\, t^2 + \cdots + \mathbf{v}_n\, t^n \tag{1.16b}$$

where $\mathbf{u}_0$ and $\mathbf{v}_0$ are the known initial displacement and velocity vectors, and $\mathbf{u}_1, ..., \mathbf{u}_1$ and $\mathbf{v}_1, ..., \mathbf{v}_1$ are the unknown coefficient vectors to be determined. By using Eqs. (1.14a) and (1.14b), two sets of weighted residual statements were defined as

$$\int_{t_s}^{t_{s+1}} (\bar{\mathbf{v}}(t) + a\ \dot{\tilde{\mathbf{v}}}(t))^T \left(\mathbf{M}\dot{\tilde{\mathbf{v}}}(t) + \mathbf{C}\tilde{\mathbf{v}}(t) + \mathbf{K}\tilde{\mathbf{u}}(t) - \mathbf{f}(t)\right) \mathrm{dt} = \mathbf{0} \tag{1.17a}$$

$$\int_{t_s}^{t_{s+1}} (\bar{\mathbf{u}}(t) + a\ \dot{\bar{\mathbf{u}}}(t))^T \left(\tilde{\mathbf{v}}(t) - \dot{\tilde{\mathbf{u}}}(t)\right) \mathrm{dt} = \mathbf{0} \tag{1.17b}$$

where

$$\bar{\mathbf{u}}(t) = \bar{\mathbf{u}}_1\, t + \bar{\mathbf{u}}_2\, t^2 + \cdots + \bar{\mathbf{u}}_n\, t^n \tag{1.18a}$$

$$\bar{\mathbf{v}}(t) = \bar{\mathbf{v}}_1\, t + \bar{\mathbf{v}}_2\, t^2 + \cdots + \bar{\mathbf{v}}_n\, t^n \tag{1.18b}$$

Here a is an algorithmic parameter used for the control of algorithmic dissipation. In Idesman's original paper [43], a was restated in terms of the dimensionless parameter α as $a = \Delta t/\alpha$. Since Eq. (1.17) should be integrated over the time interval ($t_s \leq t \leq t_{s+1}$), it is difficult to apply the algorithms to nonlinear analyses.

Fung[60] also used the weighted residual method, but he directly manipulated the equation of structural dynamics given in Eq. (1.1). The weighted residual statement considered by Fung can be written as

$$\int_{t_s}^{t_{s+1}} w_i(t)\ \left(\mathbf{M}\ddot{\tilde{\mathbf{u}}}(t) + \mathbf{C}\dot{\tilde{\mathbf{u}}}(t) + \mathbf{K}\tilde{\mathbf{u}}(t) - \mathbf{f}(t)\right) \mathrm{d}t = \mathbf{0} \quad \text{for} \quad i = 1, 2, ..., n \tag{1.19}$$

where $\tilde{\mathbf{u}}(t)$ is the approximation of $\mathbf{u}(\mathbf{t})$, and $w_i(t)$ is the ith weight function. In Fung's weighted residual approach, $\tilde{\mathbf{u}}(t)$ was specially constructed to satisfy the initial conditions given in Eqs. (1.2a) and (1.2b). In Fung's work, the displacement vector was approximated as

$$\tilde{\mathbf{u}}(t) = \mathbf{u}_0 + \mathbf{v}_0\, t + \mathbf{u}_2\, t^2 + \mathbf{u}_3\, t^3 + \cdots + \mathbf{u}_{n+1}\, t^{n+1} \tag{1.20}$$

where $\mathbf{u}_0$ and $\mathbf{v}_0$ are the initial displacement and velocity vectors, and $\mathbf{u}_2, \mathbf{u}_3, ..., \mathbf{u}_{n+1}$ are the unknown coefficient vectors. The weighted residual approach used by Fung was advantageous because the weight parameters [59, 79] were used to rewrite the integral forms of the weighted residual statement given in Eq. (1.19) as the algebraic form. According to Ref. [16], the weight parameters can be defined by

$$\theta_k = \frac{\int_0^{\Delta t}\ w(\tau)\ \tau^k\ \mathrm{d}\tau}{\Delta t^{\,k}\ \int_0^{\Delta t}\ w(\tau)\ \mathrm{d}\tau} \quad \text{for} \quad k = 0, 1, 2, ..., n+1 \tag{1.21}$$

where $\tau = t - t_s$ is the local time parameter. After restating the integral form of weighted residual statement given in Eq. (1.19) as algebraic forms by using the weight parameters, the weight parameters are optimized to achieve improved accuracy and stability. Since the equation of structural dynamics given in Eq. (1.1) was directly used to state the weighted residual statement, the result algorithms obtained from Fung's method cannot be applied to nonlinear analyses either.

1.2.2 Differential Quadrature Method

Unlike the existing weighted residual method based algorithms, the differential quadrature method [80] based algorithms can be applied to nonlinear and linear problems without any limitation. The accuracy of the differential quadrature method is mainly determined by the choice of the sampling points (also called the quadrature points) within the time interval. For example, in the differential quadrature method, the first order time derivative of the displacement vector at the sampling points can be stated as

$$\begin{Bmatrix} \dot{\mathbf{u}}_1 \\ \vdots \\ \dot{\mathbf{u}}_n \end{Bmatrix} = \begin{bmatrix} a_{11}\,\mathbf{I} & \cdots & a_{1n}\,\mathbf{I} \\ \vdots & & \vdots \\ a_{n1}\,\mathbf{I} & \cdots & a_{nn}\,\mathbf{I} \end{bmatrix} \begin{Bmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_n \end{Bmatrix} + \begin{bmatrix} a_{10}\,\mathbf{I} \\ \vdots \\ a_{n0}\,\mathbf{I} \end{bmatrix} \mathbf{u}_0 \tag{1.22}$$

where $\mathbf{u}_i$ is the displacement vector of size m associated with the ith sampling point t_i, m being the size of the semi-discrete system in Eq. (1.1), and $\mathbf{I}$ is an $m \times m$ identity matrix. Here, a_{ij} are the weighting coefficients which can be constructed by using known test function $\Psi(t)$ and properly chosen sampling points within the time interval. In the differential quadrature method, $n+1$ sampling points (i.e., t_i, for $i = 0, 1, 2, ..., n$) are located in the time interval, and the first sampling point (t_0) is always chosen to be the beginning of the time interval (t_s), whereas the last sampling

point (t_n) may not match the end of the time interval. By using $\Psi(t)$, the function values and their first-order time derivative at sampling points can be related as

$$\left.\frac{\mathrm{d}}{\mathrm{d}t}\Psi(t)\right|_{t=t_i} = \sum_{j=0}^{n} a_{ij}\Psi(t_j) \quad \text{for} \quad i = 0, 1, 2, ..., n \tag{1.23}$$

As the simplest case, $\Psi(t)$ can be chosen as $1, t, t^2, ...,$ or t^n. Then, a_{ij} can be constructed systematically from Eq. (1.23) since $\left.\frac{\mathrm{d}}{\mathrm{d}t}\Psi(t)\right|_{t=t_i}$ and $\Psi(t_j)$ are all known values if proper sampling points are provided. If $\Psi(t) \in \{1, t, t^2, ..., t^n\}$, a_{ij} can be constructed as

$$\begin{bmatrix} a_{00} & a_{01} & \cdots & a_{0n} \\ a_{10} & a_{11} & \cdots & a_{1n} \\ \vdots & \vdots & & \vdots \\ a_{n0} & a_{n1} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & t_0 & \cdots & t_0^n \\ 1 & t_1 & \cdots & t_1^n \\ \vdots & \vdots & & \vdots \\ 1 & t_n & \cdots & t_n^n \end{bmatrix} \begin{bmatrix} 0 & 1 & & \\ & 0 & \ddots & \\ & & \ddots & n \\ & & & 0 \end{bmatrix} \begin{bmatrix} 1 & t_0 & \cdots & t_0^n \\ 1 & t_1 & \cdots & t_1^n \\ \vdots & \vdots & & \vdots \\ 1 & t_n & \cdots & t_n^n \end{bmatrix}^{-1} \tag{1.24}$$

In the traditional differential quadrature methods, both equally spaced sampling points and some specially spaced points (such as the Chebyshev-Gauss-Lobatto and the Legendre-Chebyshev points) were used. However, in the modified differential quadrature method considered by Fung [69, 44], the sampling points are carefully chosen to achieve improved accuracy, unconditional stability and controllable algorithmic dissipation. Similarly, the discrete acceleration-velocity relation is stated by using a_{ij} as follows:

$$\begin{Bmatrix} \ddot{\mathbf{u}}_1 \\ \vdots \\ \ddot{\mathbf{u}}_n \end{Bmatrix} = \begin{bmatrix} a_{11}\,\mathbf{I} & \cdots & a_{1n}\,\mathbf{I} \\ \vdots & & \vdots \\ a_{n1}\,\mathbf{I} & \cdots & a_{nn}\,\mathbf{I} \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{u}}_1 \\ \vdots \\ \dot{\mathbf{u}}_n \end{Bmatrix} + \begin{bmatrix} a_{10}\,\mathbf{I} \\ \vdots \\ a_{n0}\,\mathbf{I} \end{bmatrix} \dot{\mathbf{u}}_0 \tag{1.25}$$

To state accelerations vectors in terms of displacement vectors, $\dot{\mathbf{u}}_i$ in Eq. (1.25) are eliminated by using the relation given in Eq. (1.22). In Fung's modified differential quadrature method, $\ddot{\mathbf{u}}_i$ can be stated as

$$\begin{Bmatrix} \ddot{\mathbf{u}}_1 \\ \vdots \\ \ddot{\mathbf{u}}_n \end{Bmatrix} = \begin{bmatrix} \bar{a}_{11}\,\mathbf{I} & \cdots & \bar{a}_{1n}\,\mathbf{I} \\ \vdots & & \vdots \\ \bar{a}_{n1}\,\mathbf{I} & \cdots & \bar{a}_{nn}\,\mathbf{I} \end{bmatrix} \begin{Bmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_n \end{Bmatrix} + \begin{bmatrix} \bar{a}_{10}\,\mathbf{I} \\ \vdots \\ \bar{a}_{n0}\,\mathbf{I} \end{bmatrix} \mathbf{u}_0 + \begin{bmatrix} a_{10}\,\mathbf{I} \\ \vdots \\ a_{n0}\,\mathbf{I} \end{bmatrix} \mathbf{v}_0 \tag{1.26}$$

where $\bar{a}_{ij}$ and $\bar{a}_{i0}$ are computed as,

$$\begin{bmatrix} \bar{a}_{11}\,\mathbf{I} & \cdots & \bar{a}_{1n}\,\mathbf{I} \\ \vdots & & \vdots \\ \bar{a}_{n1}\,\mathbf{I} & \cdots & \bar{a}_{nn}\,\mathbf{I} \end{bmatrix} = \begin{bmatrix} a_{11}\,\mathbf{I} & \cdots & a_{1n}\,\mathbf{I} \\ \vdots & & \vdots \\ a_{n1}\,\mathbf{I} & \cdots & a_{nn}\,\mathbf{I} \end{bmatrix} \begin{bmatrix} a_{11}\,\mathbf{I} & \cdots & a_{1n}\,\mathbf{I} \\ \vdots & & \vdots \\ a_{n1}\,\mathbf{I} & \cdots & a_{nn}\,\mathbf{I} \end{bmatrix} \tag{1.27a}$$

$$\begin{bmatrix} \bar{a}_{10}\,\mathbf{I} \\ \vdots \\ \bar{a}_{n0}\,\mathbf{I} \end{bmatrix} = \begin{bmatrix} a_{11}\,\mathbf{I} & \cdots & a_{1n}\,\mathbf{I} \\ \vdots & & \vdots \\ a_{n1}\,\mathbf{I} & \cdots & a_{nn}\,\mathbf{I} \end{bmatrix} \begin{bmatrix} a_{10}\,\mathbf{I} \\ \vdots \\ a_{n0}\,\mathbf{I} \end{bmatrix} \tag{1.27b}$$

In the differential quadrature method, n dynamic equilibrium equations are required to find n unknown displacement vectors associated with the corresponding n sampling points. To apply the relations given in Eqs. (1.22) and (1.26), the n dynamic equilibrium equations can be written in the matrix form as

$$\begin{bmatrix} \mathbf{M} & & \\ & \ddots & \\ & & \mathbf{M} \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{u}}_1 \\ \vdots \\ \ddot{\mathbf{u}}_n \end{Bmatrix} + \begin{bmatrix} \mathbf{C} & & \\ & \ddots & \\ & & \mathbf{C} \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{u}}_1 \\ \vdots \\ \dot{\mathbf{u}}_n \end{Bmatrix} + \begin{bmatrix} \mathbf{K} & & \\ & \ddots & \\ & & \mathbf{K} \end{bmatrix} \begin{Bmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_n \end{Bmatrix} = \begin{Bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_n \end{Bmatrix} \tag{1.28}$$

Substitution of Eqs. (1.22) and (1.26) into Eq. (1.28) can be simplified as

$$\left(\begin{bmatrix} \bar{a}_{11}\mathbf{M} & \cdots & \bar{a}_{1n}\mathbf{M} \\ \vdots & \vdots & \vdots \\ \bar{a}_{n1}\mathbf{M} & \cdots & \bar{a}_{nn}\mathbf{M} \end{bmatrix} + \begin{bmatrix} a_{11}\mathbf{C} & \cdots & a_{1n}\mathbf{C} \\ \vdots & \vdots & \vdots \\ a_{n1}\mathbf{C} & \cdots & a_{nn}\mathbf{C} \end{bmatrix} + \begin{bmatrix} \mathbf{K} & & \\ & \ddots & \\ & & \mathbf{K} \end{bmatrix} \right) \begin{Bmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_n \end{Bmatrix} = \begin{Bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_n \end{Bmatrix} - \begin{bmatrix} \bar{a}_{10}\,\mathbf{M} + \bar{a}_{10}\,\mathbf{C} \\ \vdots \\ \bar{a}_{n0}\,\mathbf{M} + \bar{a}_{n0}\,\mathbf{C} \end{bmatrix} \mathbf{u}_0 - \begin{bmatrix} \bar{a}_{10}\,\mathbf{M} \\ \vdots \\ \bar{a}_{n0}\,\mathbf{M} \end{bmatrix} \mathbf{v}_0 \tag{1.29}$$

Then $\mathbf{u}_i$ can be found by solving Eq. (1.29).

As shown in Eqs. (1.22) - (1.29), the application of the differential quadrature method to the linear structural dynamics problems is very simple and intuitive, once the weighting coefficients are properly constructed. However, improved accuracy, unconditional stability and algorithmic dissipation control are not provided in the conventional differential quadrature method. Only the modified differential quadrature method considered by Fung can be used to develop higher-order algorithms with improved accuracy, unconditional stability and algorithmic dissipation control. In Fung's method, the sampling points (the quadrature points) have been optimized by using the exact solution of the homogeneous single degree of freedom problem to achieve higher-order accuracy, the dissipation control capability, and the unconditional stability.

1.2.3 Some Limitations of Existing Higher-Order Algorithms

Among many of the existing weighted residual method based higher-order algorithms, the algorithms of Fung [60] and Idesman [43] can control numerical dissipations of the algorithms. However the algorithms of Idesman can provide only limited range of numerical dissipations in the high frequency limit. In both weighted residual

methods, time dependent variables have been approximated as polynomial expressions satisfying given initial conditions as presented in Eq. (1.16) and (1.20). In these methods, the approximation coefficient vectors do not have physical meanings. As a result, numerical solutions should be computed at the end of computation by using the approximations after determining the approximation coefficient vectors. In addition, algorithms proposed by Idesman can only provide less accurate solutions compared to Fung's algorithms. In our own review of these two algorithms, we found that only nth-order accuracy could be obtained with the algorithms proposed by Idesman. On the other hand, the algorithms of Fung could provide $(2n-1)$ th-order accuracy.

The modified differential quadrature method considered by Fung can provide higher-order algorithms with improved accuracy, unconditional stability, and controllable numerical dissipation. However, the sampling points of Fung's differential quadrature method are quite different from those of the traditional differential quadrature method. In Fung's method, the first sampling point is always chosen to be t_s, which is the same in the traditional quadrature method. However the last sampling point does not always match t_{s+1}. For example, in the non-dissipative case of the 4th-order algorithm ($n = 2$) obtained by using Fung's differential quadrature method, three quadrature points should be chosen as $t_0 = 0.0$, $t_1 = 0.2113248653$, and $t_2 = 0.7886751347$. In Fung's method, the first sampling point always matches the beginning of the time interval. In this particular case, $\mathbf{u}_0$ and $\mathbf{v}_0$ are known values at $t = t_0\Delta t$, and displacement solutions ($\mathbf{u}_1$ and $\mathbf{u}_2$) at $t = t_1\Delta t$ and $t_2\Delta t$ can be obtained by solving Eq. (1.29). Since t_2 is not 1.0, $\mathbf{u}_2$ of Eq. (1.29) is not $\mathbf{u}_{s+1}$. In the Fung's method, $\mathbf{u}_{s+1}$ should be reconstructed by interpolating $\mathbf{u}_0$, $\mathbf{u}_1$ and $\mathbf{u}_2$. This procedure may not increase the computational cost noticeably, but the implementation of this procedure is additional work, which is not required in other

methods.

Another additional procedure required in the Fung's method is the determination of the n sampling points for each specification of dissipation level. The n sampling points are the roots of nth-degree polynomial equation whose coefficients are functions of a free parameter which is selected by user for the control of algorithmic dissipations. The weighting parameters can be constructed after determining n sampling points. If $n = 2$, the velocity-displacement relations at two sampling points can be written down analytically in terms of the dissipation control parameter. If $n = 3, 4$, it is still possible to directly state the velocity-displacement relations at sampling points in terms of the dissipation control parameter, while the expression may become longer and more complicated than the case of $n = 2$. However, finding roots of the polynomial equation of 5th or higher degree becomes difficult, because it has been proven that roots of 5th or higher degree polynomial equation cannot be found analytically according to the Abel−Ruffini theorem. Thus, additional numerical algorithms should be used to find n roots of nth degree polynomial equation for every specification of the dissipation control parameter in Fung's method.

1.3 Motivation and Objectives

The first goal of this study is to develop a new family of second-order time integration algorithms which can overcome the limitations of the existing second-order algorithms. Preferable attributes of second-order time integration algorithms include

(1) Unconditional stability for linear problems

(2) Controllable algorithmic dissipation

(3) Second-order accuracy for dissipative cases

(4) Easy extension to nonlinear problems

attribute	Newmark	Generalized-α	Baig and Bathe
(1)	yes	yes	yes
(2)	limited	yes	no
(3)	no	yes	yes
(4)	yes	limited [46]	yes

Table 1.1: Evaluation of existing second-order algorithms by preferable attributes. (1): Unconditional stability for linear problems, (2): Controllable algorithmic dissipation, (3): Second-order accuracy for dissipative cases, (4): Easy extension to nonlinear problems

Among many of the existing second-order algorithms, only the generalized-α method can provide all attributes stated in (1)-(4). However, (4) is not fully provided in the generalized-α method as discussed in Ref. [81]. Another shortcoming of the generalized-α method is that it may introduce excessive algorithmic damping into the important low frequency range, if highly dissipative case is used. If fast high frequency filtering is required in the analysis, the Baig and Bathe method can provide more accurate low frequency solutions than the generalized-α method does. However, the Baig and Bathe method can perform as the asymptotic annihilation case only. As a result, it is not suitable for the long-term energy conserving type of problems.

As the first part of our study, we propose a new second-order algorithm developed based on (a) the Lagrange interpolation functions in time, (b) two residuals defined from unconventionally rewritten first-order equations, and (c) the collocation method to overcome the shortcomings of the two second-order algorithms in a moderate way.

As discussed previously, however, the fundamental limitations of second-order algorithms cannot be overcome without considering higher-order algorithms. Thus, the ultimate goal of this study is to provide improved higher-order time integration

algorithms that can be used for the analysis of structural dynamics in ready-to-use forms. Preferable attributes of higher-order time integration algorithms can be summarized as

(1) Unconditional stability for linear problems

(2) Controllable algorithmic dissipation

(3) Exclusion of any undetermined algorithmic parameter

(4) Exclusion of any reconstruction of solutions

(5) Conciseness of final result equation

(6) Easy application to nonlinear problems

(7) Improved $(2n-1)$ th- or $(2n)$ th-order accuracy (n being the number of unknown vectors)

Many higher-order time integration algorithms have been proposed for the effective analysis of structural dynamic problems, but none of them successfully achieved all of the preferable attributes listed above. To summarize the characteristics of the existing higher-order algorithms, their are evaluated according to the preferable attributes and the results are presented in Table 1.2.

In this study we wish to develop new higher-order algorithms which can eliminate the limitations of the existing higher-order algorithms. To design new time integration algorithms, two modified time finite elements approaches have been considered.

As the second part of our study, we propose a time finite element procedure based on (a) the Hermite interpolation functions in time, (b) the residual defined from original second-order structural dynamics equation, (c) the modified weighted residual method and (d) the weight parameters, to overcome some limitations of the existing weighted residual method based algorithms.

As the third part of our study, we propose another time finite element procedure based on (a) the Lagrange interpolation functions in time, (b) two residu-

attribute	DQM[44]	WRM[60]	WRM-TCG[43]	WRM-TDG[36]
(1)	yes	yes	limited	yes
(2)	yes	yes	limited	no
(3)	limited	yes	yes	yes
(4)	limited	limited	limited	yes
(5)	limited	limited	limited	yes
(6)	yes	no	no	no
(7)	yes	yes	limited	yes

Table 1.2: Evaluation of existing higher-order algorithms by preferable attributes. (1): Unconditional stability for linear problems, (2): Controllable algorithmic dissipation, (3): Exclusion of any undetermined algorithmic parameter, (4): Exclusion of any reconstruction of solutions, (5): Conciseness of final result equation , (6): Easy application to nonlinear problems, (7): Improved $(2n-1)$ th- or $(2n)$ th-order accuracy (n being the number of unknown vectors)

als defined from unconventionally rewritten first-order equations, (c) the modified weighted residual method and (d) the weight parameters to overcome all limitations of the differential quadrature method proposed by Fung.

1.4 Overview

In Chapter 2, a new family of implicit second-order time integration algorithm is developed, analysed, and tested. The algorithm is fully extended to nonlinear problems. The algorithm developed in this Chapter can provide better efficiency compared with existing second-order algorithms.

In Chapter 3, a new family of implicit higher-order time integration algorithms is developed, analysed, and tested. The algorithms developed in this Chapter can provide better efficiency compared with existing weighted residual method based higher-order algorithms, elimination their limitations. The algorithms presented in this chapter have very unique computational structure that improves the efficiency

of equation solving when properly implemented into computer code.

In Chapter 4, another new family of implicit higher-order time integration algorithms is developed, analysed, and tested. The algorithms are fully extended to nonlinear cases, and specific linear and nonlinear equation solving procedure is provided. The algorithms presented in this chapter can be applied to any order of initial value problems as the differential quadrature method.

In Chapter 5, conclusions are presented along with future works.

2. TIME FINITE ELEMENT METHOD I

2.1 Introduction

For many decades, an important part of linear and nonlinear dynamic analyses of structural problems has been the development of effective step-by-step implicit time integration algorithms[9, 23, 24, 26] that are robust and efficient (i.e., stable and accurate). Several successful algorithms[20, 27, 34, 7] have been developed based on modified structural dynamics equations and the Newmark scheme[23]. In these algorithms, the algorithmic dissipation has been effectively controlled by utilizing the numerical damping caused by the modification of structural dynamics equations, and the parameters of the Newmark scheme also have been selected correspondingly to maintain accuracy and stability. In the modified structural dynamics equation, different evaluation points (in time) of forces (i.e., inertia, damping, internal resistance, and externally applied forces) serve as the main mechanism of the algorithmic numerical damping, which can improve the quality of numerical solutions with the appropriate use of the parameters of the Newmark scheme.

The generalized-α method[7] can be viewed as one belonging to this category. Among four parameters of the generalized-α method, two parameters are the original parameters of the Newmark scheme[23] and the others, called alpha parameters, are introduced to determine the points in the time domain where the forces of the structural dynamics equation are evaluated. The most distinct feature of the generalized-α method from several previously developed alpha type methods[20, 49, 34] which use

*Reprinted with permission from An Improved Time Integration Algorithm: A Collocation Time Finite Element Approach by Wooram Kim and J.N.Reddy, 2016. International Journal of Structural Stability and Dynamics, Copyright [2016] by World Scientific.

only one additional parameter, is that the generalized-α method uses two additional parameters to evaluate the inertia and other type of forces in the structural dynamics equation at two adjacent equilibrium points. By optimizing all four parameters, the generalized-α method can retain the second-order accuracy for any dissipative case.

In general, among infinite numbers of dissipative cases which can be obtained from any time integration algorithm, the asymptotic annihilation and no-dissipation cases are most important in practical analyses. Especially, the asymptotic annihilation case has a very special property that can eliminate any artifact that comes from the high frequency range within one time step. Usually, the artifact of the high frequency is due to poor representations of spatial domain in numerical methods. However, the asymptotic annihilation case obtained from the generalized-α method shows a large (period) error and becomes too dissipative even with considerably small time step. Some of the traditionally developed asymptotic annihilation algorithms, such as the Houbolt and Park methods, are designed to perform as the asymptotic annihilation case only. Especially, the Park method is not too dissipative in practical low frequency ranges when it is compared with the asymptotic annihilation case of the generalized-α method. Naturally, considerably small time step should be used in the the asymptotic annihilation case of the generalized-α method to match the performance of some accurate traditional asymptotic annihilation algorithms, thus requiring increased computational expense.

A totally different approach has been taken by Baig and Bathe [35, 40], where an effective and unconditionally stable asymptotic annihilation time integration algorithm was developed by simply combining two existing well-known time integration schemes. In this method, the time interval was divided into two sub-steps. For the first sub-step, the one-step average constant acceleration case of the Newmark scheme (also called the trapezoidal rule) was used. For the second sub-step, the

results obtained from the first sub-step were used as the mid-point properties of the three-point Euler backward scheme. Unlike the generalized-α method, which was designed to control dissipation levels, the method of Baig and Bathe cannot control the dissipation, but rather performs only as an asymptotic annihilation scheme. However, the Baig and Bathe method shows much better performance compared with the asymptotic annihilation case of the generalized-α method. It can effectively eliminate high frequency effects introducing very small algorithmic damping into low frequency range. However, the absence of the dissipation control capability in any time integration algorithm can also be viewed as a handicap, while not having any specification of parameter in the Baig and Bathe method was considered a desirable feature of it by Bathe and Noh [8]. In general, a small amount of numerical damping in low frequency region can help to stabilize solutions in nonlinear dynamic systems, and dissipative time integration algorithms bring out not only prescribed level of dissipation in high frequency range but also certain amount of algorithmic damping in low frequency range. In many cases, a very small amount of algorithmic damping is enough to stabilize solutions in nonlinear dynamic systems. Since a less dissipative case can provide more accurate numerical solutions compared with the asymptotic annihilation case within a family of time integration algorithms, algorithmic dissipation should be used at the minimum if the purpose of its use is not a complete elimination of the high frequency effect.

It should be also noted that asymptotic annihilation time integration algorithms may produce very inaccurate numerical solutions even with a reasonable time step size. In general, nonlinear analysis does not allow the modal analysis and it may not be an easy task for a user to select a suitable time step size at the very beginning of the analysis. If a large time step size is chosen in the first trial, the asymptotic annihilation case may filter out some important low frequency information and the

user may not know about it. In many cases, distorted numerical solutions may look reasonably good, which can make a user stop refining the numerical solutions. One interesting and important observation is that the no-dissipation case does not eliminate any high frequency effect. However, it just includes the high frequency mode in the numerical solution with large errors. Even though the quality of the solution obtained from the no-dissipation case with large time step is not acceptable, inclusion of some spurious responses in numerical solutions can be used as an indication to refine the time step size according to our numerical experiments. If the no-dissipation case is not working in some nonlinear problems, causing instability, not only the asymptotic annihilation case but also some other less dissipative cases can be used. Thus, it is recommended that one use various dissipation levels and time step sizes even though it requires additional computational effort.

The purpose of this study is to develop a new family of time integration algorithms with dissipation control capability (like in the generalized-α method) while minimizing the algorithmic damping in low-frequency regime (i.e., embracing the idea of the Baig and Bathe method). Towards this end, we develop a new family of time integration algorithms based on the collocation finite element approach. Thus, the proposed family of algorithms is designed to possess the dissipation control capability through the collocation parameters, while adopting the computational structure of the Baig and Bathe method. This is accomplished by replacing the original second-order time differential equation of structural dynamics with an equivalent set of first-order equations and minimizing their residuals in a non-conventional setting, resulting in the one-step and two-step time integration schemes. An interesting linear spring example, which is a modification of the original example of Bathe and Noh [8], is used as an numerical example to demonstrate (a) a potential misuse of time integration algorithms, which work as only asymptotic annihilation case and

(b) some advantages of algorithms with dissipation control capability. The choice of the collocation points within the time interval serve as the main mechanism for the dissipation control in the new algorithm.

2.2 Development

2.2.1 Mixed Model and Related Concepts

Various types of numerical methods can be employed to spatially discretize partial differential equations (PDEs). After applying the spatial discretization on original PDEs associated with structural dynamics, the resulting ordinary differential equations (ODEs) in time are of the form

$$\mathbf{M}\ddot{\mathbf{u}}(t) + \mathbf{C}\dot{\mathbf{u}}(t) + \mathbf{K}(t)\,\mathbf{u}(t) = \mathbf{f}(t) \tag{2.1}$$

with given initial conditions

$$\mathbf{u}(0) = \mathbf{u}_0 \tag{2.2a}$$

$$\dot{\mathbf{u}}(0) = \mathbf{v}_0 \tag{2.2b}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{C}$ is the damping matrix, $\mathbf{K}(t)$ is the nonlinear stiffness matrix, $\mathbf{f}$ is the force vector, and $\mathbf{u}$ is the displacement vector. Note that $\mathbf{K}(t)\,\mathbf{u}(t)$ is used in place of the general nonlinear internal force vector, since we are assuming that the nonlinear internal force vector is linearized from the spatial discretization. Thus the nonlinearity of internal force is included in the stiffness matrix as $\mathbf{K}(t) = \mathbf{K}(\mathbf{u}(t))$. For the linear case, constat $\mathbf{K}$ can be used in place of $\mathbf{K}(t)$. Eq. (2.1) can

be expressed as a set of the following lower-order equations:

$$\mathbf{M}\mathbf{a}(t) + \mathbf{C}\mathbf{v}(t) + \mathbf{K}(t)\,\mathbf{u}(t) = \mathbf{f}(t) \tag{2.3a}$$

$$\mathbf{v}(t) = \dot{\mathbf{u}}(t) \tag{2.3b}$$

$$\mathbf{a}(t) = \dot{\mathbf{v}}(t) \tag{2.3c}$$

where $\mathbf{v}$ is the velocity vector and $\mathbf{a}$ is the acceleration vector. It is advantageous to rewrite Eq. (2.1) as a set of lower-order equations given in Eqs. (2.3a)-(2.3c) for following reason: Eq. (2.3a) is already an algebraic equation valid for any time; this will allow us to set the force equilibrium at any desired time. Equations (2.3b) and (2.3c) can be converted to algebraic equations through a residual minimization procedure in contrast to directly manipulating Eq. (2.1) to obtain the algebraic equations through some approximation procedure (e.g., using time-approximations schemes or the finite element method in time). If Eq. (2.1) is directly approximated in time using the finite element procedure based on a weighted-residual method, higher-order approximation of the displacement vector is required and the equation is satisfied only in an integral sense. By introducing the velocity and acceleration vectors as dependent variables, we can apply the finite element procedure in time on Eqs. (2.3b) and (2.3c) using lower-order approximations and use Eq. (2.3a) only for satisfying the equilibrium requirement. Due to the fact that $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{a}$ have different units, any computational model based on the use of the set in Eqs. (2.3a)-(2.3c) is termed a *mixed model* [82, 77, 83].

Based on the mixed formulation, time finite element models can be developed over a time (finite element) interval $t_s \leq t \leq t_{s+1}$, where the size of the time interval is $\Delta t = t_{s+1} - t_s$. Strictly speaking, from consistency considerations, $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{a}$ should

be interpolated with different polynomial orders (i.e., if $\mathbf{a}$ is approximated using a pth degree polynomial, then $\mathbf{v}$ and $\mathbf{u}$ should be approximated using $(p+1)$th and $(p+2)$th degree polynomials, respectively). However $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{a}$ are independently approximated by using equal lower order of interpolations and treated as nodal values of the time finite element in our study. And the inconsistency caused due to equal lower order approximations of all three variables are utilized to control dissipation.

If it is a nonlinear problem, exclusion of any time dependent coefficient (e.g., $\mathbf{K}(t)$) in the integral statement will preserve the simplicity in the final form of the algorithm, which can be easily implemented on a computer. The Newmark scheme is a good example of this type of equilibrium setting. In the Newmark scheme[23] the equilibrium is considered at $t = t_{s+1}$ (i.e., $t = t_s + \Delta t$), that is, Eq. (2.1) is evaluated at time $t = t_{s+1}$:

$$\mathbf{M}\ddot{\mathbf{u}}_{s+1} + \mathbf{C}\dot{\mathbf{u}}_{s+1} + \mathbf{K}_{s+1}\mathbf{u}_{s+1} = \mathbf{f}_{s+1} \tag{2.4}$$

Here vectors and matrices with subscript $s+1$ denote that they are functions of time and are evaluated at time $t = t_{s+1}$. Then the Newmark scheme uses truncated Taylor's series, which is given as

$$\mathbf{u}_{s+1} = \mathbf{u}_s + \Delta t\dot{\mathbf{u}}_s + \left(\frac{1}{2} - \beta\right)(\Delta t)^2\ddot{\mathbf{u}}_s + \beta(\Delta t)^2\ddot{\mathbf{u}}_{s+1} \tag{2.5a}$$

$$\dot{\mathbf{u}}_{s+1} = \dot{\mathbf{u}}_s + (1-\gamma)\,\Delta t\ddot{\mathbf{u}}_s + \gamma\Delta t\ddot{\mathbf{u}}_{s+1} \tag{2.5b}$$

where β and γ are the two truncation parameters. Expressions given in Eqs. (2.5a) and (2.5b) are used to eliminate $\dot{\mathbf{u}}_{s+1}$ and $\ddot{\mathbf{u}}_{s+1}$ in Eq. (2.4) and the unknown vector $\mathbf{u}_{s+1}$ is expressed in terms of $\mathbf{u}_s$ and known mechanical, inertial, and damping forces

as follows (see Reddy[1]; assuming that the mass and damping matrices are not functions of $\mathbf{u}$):

$$\hat{\mathbf{K}}_{s+1}\mathbf{u}_{s+1} = \hat{\mathbf{F}}_{s,s+1} \tag{2.6}$$

where

$$\begin{aligned} \hat{\mathbf{K}}_{s+1} &= \mathbf{K}_{s+1} + a_3\mathbf{M} + a_6\mathbf{C} \\ \hat{\mathbf{F}}_{s,s+1} &= \mathbf{F}_{s+1} + \mathbf{F}_s^{\mathrm{I}} + \mathbf{F}_s^{\mathrm{D}} \\ \mathbf{F}_s^{\mathrm{I}} &= \mathbf{M}\left(a_3\mathbf{u}_s + a_4\dot{\mathbf{u}}_s + a_5\ddot{\mathbf{u}}_s\right) \\ \mathbf{F}_s^{\mathrm{D}} &= \mathbf{C}\left(a_6\mathbf{u}_s + a_7\dot{\mathbf{u}}_s + a_8\ddot{\mathbf{u}}_s\right) \end{aligned} \tag{2.7}$$

and

$$\begin{aligned} a_3 = \frac{1}{\beta(\Delta t)^2}, \quad a_4 = a_3\Delta t, \quad a_5 = \frac{1}{2\beta} - 1 \\ a_6 = \frac{\gamma}{\beta\Delta t}, \quad a_7 = \frac{\gamma}{\beta} - 1, \quad a_8 = \left(\frac{\gamma}{2\beta} - 1\right)\Delta t \end{aligned} \tag{2.8}$$

After obtaining $\mathbf{u}_{s+1}$, $\ddot{\mathbf{u}}_{s+1}$ and $\dot{\mathbf{u}}_{s+1}$ are updated as

$$\begin{aligned} \ddot{\mathbf{u}}_{s+1} &= a_3(\mathbf{u}_{s+1} - \mathbf{u}_s) - a_4\dot{\mathbf{u}}_s - a_5\ddot{\mathbf{u}}_s \\ \dot{\mathbf{u}}_{s+1} &= \dot{\mathbf{u}}_s + a_2\ddot{\mathbf{u}}_s + a_1\ddot{\mathbf{u}}_{s+1} \end{aligned} \tag{2.9}$$

where

$$a_1 = \gamma\Delta t, \quad a_2 = (1-\gamma)\Delta t \tag{2.10}$$

Some "improved schemes" based on Eqs. (2.5a) and (2.5b) have been proposed in some what ad-hoc way ("end justifies the means" approach). For example, the HHT-α method of Hilber, Hughes, and Taylor[20] considered the equilibrium of the

following equation in lieu of Eq. (2.4):

$$\begin{aligned}&\mathbf{M}\ddot{\mathbf{u}}_{s+1} + (1+\alpha_h)\,\mathbf{C}\dot{\mathbf{u}}_{s+1} - \alpha_h \mathbf{C}\dot{\mathbf{u}}_s \\ &+ (1+\alpha_h)\,\mathbf{K}_{s+1}\mathbf{u}_{s+1} - \alpha_h \mathbf{K}_s \mathbf{u}_s = (1+\alpha_h)\,\mathbf{f}_{s+1} - \alpha_h \mathbf{f}_s\end{aligned} \tag{2.11}$$

where α_h is a parameter whose value is chosen to control certain numerical response of the scheme. In a similar way, the WBZ-α method[34] considered the equilibrium of

$$(1-\alpha_b)\,\mathbf{M}\ddot{\mathbf{u}}_{s+1} + \alpha_b \mathbf{M}\ddot{\mathbf{u}}_s + \mathbf{C}\dot{\mathbf{u}}_{s+1} + \mathbf{K}_{s+1}\mathbf{u}_{s+1} = \mathbf{f}_{s+1} \tag{2.12}$$

where α_b is an additional parameter. We note that the inertia term $\mathbf{M}\ddot{\mathbf{u}}_{s+1}$ of Eq. (2.4) is evaluated as $(1-\alpha_b)\,\mathbf{M}\ddot{\mathbf{u}}_{s+1} + \alpha_b \mathbf{M}\ddot{\mathbf{u}}_s$. Finally, the generalized-α method [7, 45, 81] considered the equilibrium of

$$\begin{aligned}&(1-\alpha_m)\,\mathbf{M}\ddot{\mathbf{u}}_{s+1} + \alpha_m \mathbf{M}\ddot{\mathbf{u}}_s + (1-\alpha_f)\,\mathbf{C}\dot{\mathbf{u}}_{s+1} + \alpha_f \mathbf{C}\dot{\mathbf{u}}_s \\ &+ (1-\alpha_f)\,\mathbf{K}_{s+1}\mathbf{u}_{s+1} + \alpha_f \mathbf{K}_s \mathbf{u}_s = (1-\alpha_f)\,\mathbf{f}_{s+1} + \alpha_f \mathbf{f}_s\end{aligned} \tag{2.13}$$

where α_m and α_f are two independent parameters. The generalized-α method includes the HHT-α and the WBZ-α method as special cases (set $\alpha_m = 0$ and $\alpha_f = -\alpha_h$ to obtain the HHT-α method and $\alpha_m = \alpha_b$ and $\alpha_f = 0$ to obtain the WBZ-α method). In all three methods, the Newmark approximations given in Eqs. (2.5a) and (2.5b) are used to eliminate $\dot{\mathbf{u}}_{s+1}$ and $\ddot{\mathbf{u}}_{s+1}$, and to obtain final algebraic equations for $\mathbf{u}_{s+1}$ in terms of known quantities. Note that Eqs. (2.11) and (2.13) contain stiffness matrix evaluated at two different times, and they should be tracked separately during iterations in a nonlinear analysis.

The original paper by Hilber, Hughes, and Taylor[20] did not have the terms involving the damping matrix $\mathbf{C}$; Eq. (2.11) is taken from Ref. [49].

In current study we will consider equilibrium of Eq. (2.3a) at two different time steps, namely at $t = t_s + \tau\Delta t$ and $t = t_s + \Delta t$, where $0.5 \leq \tau < 1.0$, as shown in Fig. 2.1. For an arbitrary point within a time step, the equilibrium can be expressed as

$$\mathbf{M}\mathbf{a}_{s+\tau} + \mathbf{C}\mathbf{v}_{s+\tau} + \mathbf{K}_{s+\tau}\mathbf{u}_{s+\tau} = \mathbf{f}_{s+\tau} \tag{2.14}$$

In our study we will consider arbitrary equilibrium point (i.e., $t = t_s + \tau\Delta t$) for the first sub-step and fixed equilibrium point for the second sub-step (i.e., $t = t_s + \Delta t$). We note that current notion of setting the equilibrium at two different times is very similar to the collocation approach considered by Hilber and Hughes[27, 49]. The collocation approach considered by Hilber and Hughes used the same collocation points for both the equilibrium setting and Newmark's approximation. In the current case, τ is similar to the collocation parameters used in Ref. [27], but τ only determines the size of the first sub-step and the location of the second node of the second sub-step. Details are discussed in the later sections.

We now formulate our new schemes based on the time finite element approach applied to the set of first-order equations involving displacement, velocity, and acceleration vectors. To be specific, the collocation finite element model is employed to discretize Eqs. (2.3b) and (2.3c) in time with two independent collocation parameters for each sub-step. In other words, we develop a new family of time approximations based on the collocation finite element model where the collocation points are chosen judiciously, as will be discussed in detail in the sequel.

As briefly stated before, another advantage of the mixed formulation is that the evaluation of Eqs. (2.3b) and (2.3c) at a certain collocation point within the time interval after approximating variables with equal order of the Lagrange interpolation

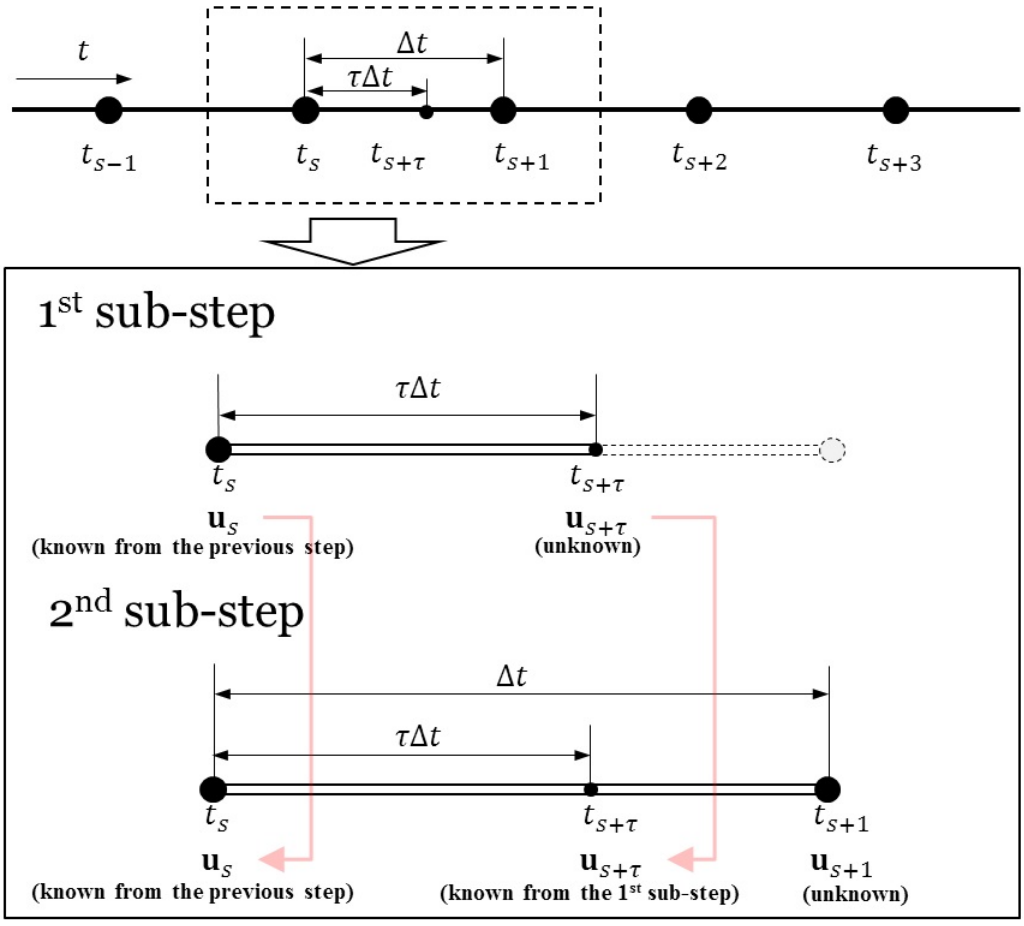

Figure 2.1: Concept of algorithm

functions can provide the dissipation control mechanism. The collocation approach can provide dissipation control mechanism to the algorithm because the level of algorithmic damping for a given size of time step changes depending on locations (i.e. collocation points) where the residuals are evaluated within the time interval. The collocation points will play a role very similar to the weight parameters used by Zienkiewicz[79, 59]. Thus the collocation point in the time finite element plays an important role in adjusting the level of algorithmic dissipation. Details of the dissipation control mechanism will be discussed at the end of the section.

2.2.2 First Sub-Step

In the Baig and Bathe method, the average acceleration method of the Newmark schemes (i.e., the trapezoidal rule) was employed for the first sub-step and the three-point Euler backward method was employed for the second sub-step. In every recurrence, properties obtained from the first sub-step were used as the second nodal properties of the second sub-step. In our study, we adopt exactly the same strategy of the Baig and Bathe method in developing new time schemes. Details of general time finite element approximations of time dependent variables can be found in the works of Oden [84], Argyris and Mlejnek [85], Hulbert [36], Fung [59] and Singh and Kalra [86]. Hulbert used the Lagrange interpolations in time to approximate the displacement and velocity vectors and applied the time discontinuous Galerkin method. Fung used the Hermite cubic interpolation functions in time to approximate the displacement vector with the weighted residual method. Also some of very fundamental concepts of time finite element approximations are well provided in Refs. [79, 84, 87]. Since we lowered the differentiability requirement on the displacement by including the velocity and acceleration vectors in the mixed formulation, we can use the same and lower order approximations for all variables.

For the first sub-step, we use the time collocation finite element approach to discretize Eqs. (2.3b) and (2.3c) over $t_s \leq t \leq t_{s+\tau}$. For the time discretization of the first sub-stpe, we use the linear interpolations of

$$\psi_s^L = \frac{\tau\Delta t - t + t_s}{\tau\Delta t}, \quad \psi_{s+\tau}^L = \frac{t - t_s}{\tau\Delta t} \tag{2.15}$$

where the superscript L is used for the linear interpolation functions. Note that τ is the same parameter used in Eq. (2.14). Here, τ adjusts the time element size, and naturally the equilibrium of the first sub-step is evaluated at the end of it. The length of the time element of the first sub-step is computed as $\tau\Delta t$ and the time at end of the first sub-step is computed as $t_{s+\tau} = t_s + \tau\Delta t$. Fig. 2.1 shows the time elements and the idea of the new algorithm schematically. Using Eq. (2.15) the variables in the first sub-step can be approximated as

$$\mathbf{u}(t) \approx \bar{\mathbf{u}}(t) = \psi_s^L(t)\,\mathbf{u}_s + \psi_{s+\tau}^L(t)\,\mathbf{u}_{s+\tau} \tag{2.16a}$$

$$\mathbf{v}(t) \approx \bar{\mathbf{v}}(t) = \psi_s^L(t)\,\mathbf{v}_s + \psi_{s+\tau}^L(t)\,\mathbf{v}_{s+\tau} \tag{2.16b}$$

$$\mathbf{a}(t) \approx \bar{\mathbf{a}}(t) = \psi_s^L(t)\,\mathbf{a}_s + \psi_{s+\tau}^L(t)\,\mathbf{a}_{s+\tau} \tag{2.16c}$$

where, $\bar{\mathbf{u}}$, $\bar{\mathbf{v}}$, and $\bar{\mathbf{a}}$ are approximated variables. Then the substitution of Eqs. (2.16a)-(2.16c) into Eqs. (2.3b) and (2.3c) gives the residuals

$$\mathbf{r}_1(t) = \bar{\mathbf{v}}(t) - \dot{\bar{\mathbf{u}}}(t) \neq \mathbf{0} \tag{2.17a}$$

$$\mathbf{r}_2(t) = \bar{\mathbf{a}}(t) - \dot{\bar{\mathbf{v}}}(t) \neq \mathbf{0} \tag{2.17b}$$

For the minimization of the residuals in Eqs. (2.17a) and (2.17b), we employ the

collocation method which can be stated in the weighted residual form as

$$\mathbf{0} = \int_{t_s}^{t_{s+\tau}} \delta\left(t - \theta_1 \tau \Delta t\right) \mathbf{r}_1 \, dt, \quad t_s \leq t \leq t_{s+\tau} \tag{2.18a}$$

$$\mathbf{0} = \int_{t_s}^{t_{s+\tau}} \delta\left(t - \theta_1 \tau \Delta t\right) \mathbf{r}_2 \, dt, \quad t_s \leq t \leq t_{s+\tau} \tag{2.18b}$$

where θ_1 is the parameter which determines the collocation point of the first sub-step. For unconditionally stable schemes, θ_1 should be chosen from the interval $0.5 \leq \theta_1 \leq 1.0$. Then $\mathbf{v}_{s+\tau}$ and $\mathbf{a}_{s+\tau}$ can be stated in term of the known properties at $t = t_s$ and $\mathbf{u}_{s+\tau}$ as

$$\mathbf{v}_{s+\tau} = c_1 \mathbf{u}_{s+\tau} + c_2 \mathbf{u}_s + c_3 \mathbf{v}_s \tag{2.19a}$$

$$\mathbf{a}_{s+\tau} = c_1 \mathbf{v}_{s+\tau} + c_2 \mathbf{v}_s + c_3 \mathbf{a}_s \tag{2.19b}$$

and the coefficients are defined as

$$c_1 = \frac{1}{\tau \theta_1 \Delta t}, \quad c_2 = -\frac{1}{\tau \theta_1 \Delta t}, \quad c_3 = \frac{\theta_1 - 1}{\theta_1} \tag{2.20}$$

As mentioned previously we consider the equilibrium at $t = t_s + \tau \Delta t$ as given in Eq. (2.14). By substituting Eqs. (2.19a) and (2.19b) into Eq. (2.14), we get the fully discretized equation of

$${}^1\hat{\mathbf{K}} \mathbf{u}_{s+\tau} = {}^1\hat{\mathbf{f}} \tag{2.21}$$

where ${}^1\hat{\mathbf{K}}$ is the effective coefficient matrix of the first sub-step, $\mathbf{u}_{s+\tau}$ is the displacement vector at $t = t_s + \tau \Delta t$, and ${}^1\hat{\mathbf{f}}$ is the effective force vector of the first sub-step.

${}^1\hat{\mathbf{K}}$ can be calculated as

$$ {}^1\hat{\mathbf{K}} = c_4\mathbf{M} + c_5\mathbf{C} + \mathbf{K}_{s+\tau} \tag{2.22} $$

where

$$ c_4 = {c_1}^2, \quad c_5 = c_1 \tag{2.23} $$

And ${}^1\hat{\mathbf{f}}$ can be calculated as

$$ {}^1\hat{\mathbf{f}} = \mathbf{M}\left(c_6\mathbf{u}_s + c_7\mathbf{v}_s + c_8\mathbf{a}_s\right) + \mathbf{C}\left(c_9\mathbf{u}_s + c_{10}\mathbf{v}_s\right) + \mathbf{f}_{s+\tau} \tag{2.24} $$

where

$$ \begin{gathered} c_6 = -c_1 c_2, \quad c_7 = -\left(c_1 c_3 + c_2\right) \\ c_8 = -c_3, \quad c_9 = -c_2, \quad c_{10} = -c_3 \end{gathered} \tag{2.25} $$

Note that each approximation of the first sub-step developed herein is equivalent to the well-known generalized trapezoidal family[6].

2.2.3 Second Sub-Step

For the second sub step, we also use the collocation finite element approach to discretize Eqs. (2.3b) and (2.3c) over $t_s \le t \le t_s + \Delta t$. Here we used the quadratic

interpolation functions

$$\begin{aligned}\psi_s^Q(t) &= \frac{(t-t_s-\tau\Delta t)(t-t_s-\Delta t)}{\tau\Delta t^2} \\ \psi_{s+\tau}^Q(t) &= \frac{(t-t_s)(t-t_s-\Delta t)}{\tau\Delta t^2(\tau-1)} \\ \psi_{s+1}^Q(t) &= \frac{(t-t_s)(t-t_s-\tau\Delta t)}{\Delta t^2(\tau-1)}\end{aligned} \tag{2.26}$$

the superscript Q is used for the quadratic interpolation functions. Using Eq. (2.26), the variables in the second sub-step can be approximated as

$$\mathbf{u}(t) \approx \bar{\mathbf{u}}(t) = \psi_s^Q(t)\,\mathbf{u}_s + \psi_{s+\tau}^Q(t)\,\mathbf{u}_{s+\tau} + \psi_{s+1}^Q(t)\,\mathbf{u}_{s+1} \tag{2.27a}$$

$$\mathbf{v}(t) \approx \bar{\mathbf{v}}(t) = \psi_s^Q(t)\,\mathbf{v}_s + \psi_{s+\tau}^Q(t)\,\mathbf{v}_{s+\tau} + \psi_{s+1}^Q(t)\,\mathbf{v}_{s+1} \tag{2.27b}$$

$$\mathbf{a}(t) \approx \bar{\mathbf{a}}(t) = \psi_s^Q(t)\,\mathbf{a}_s + \psi_{s+\tau}^Q(t)\,\mathbf{a}_{s+\tau} + \psi_{s+1}^Q(t)\,\mathbf{a}_{s+1} \tag{2.27c}$$

where, $\bar{\mathbf{u}}$, $\bar{\mathbf{v}}$, and $\bar{\mathbf{a}}$ are approximated variables. Similar to the first sub-step, the minimization of the residuals in the second sub-step can be done as

$$\mathbf{0} = \int_{t_s}^{t_{s+1}} \delta(t-\theta_2\Delta t)\,\mathbf{r}_1\,dt, \quad t_s \le t \le t_{s+1} \tag{2.28a}$$

$$\mathbf{0} = \int_{t_s}^{t_{s+1}} \delta(t-\theta_2\Delta t)\,\mathbf{r}_2\,dt, \quad t_s \le t \le t_{s+1} \tag{2.28b}$$

where θ_2 is the parameter that determines the collocation point of the second sub-step and $t_{s+1} = t_s + \Delta t$. Then $\mathbf{v}_{s+1}$ and $\mathbf{a}_{s+1}$ can be stated in term of the known properties at $t = t_s$, $t = t_{s+\tau}$ and $\mathbf{u}_{s+1}$ as

$$\mathbf{v}_{s+1} = d_1\mathbf{u}_{s+1} + d_2\mathbf{u}_{s+\tau} + d_3\mathbf{u}_s + d_4\mathbf{v}_{s+\tau} + d_5\mathbf{v}_s \tag{2.29a}$$

$$\mathbf{a}_{s+1} = d_1\mathbf{v}_{s+1} + d_2\mathbf{v}_{s+\tau} + d_3\mathbf{v}_s + d_4\mathbf{a}_{s+\tau} + d_5\mathbf{a}_s \tag{2.29b}$$

and the coefficients are defined as

$$\begin{gathered} d_1 = \frac{\tau - 2\theta_2}{\theta_2(\tau-\theta_2)\Delta t}, \quad d_2 = \frac{2\theta_2 - 1}{\tau\theta_2(\tau-\theta_2)\Delta t} \\ d_3 = \frac{(1-\tau)(\tau+1-2\theta_2)}{\tau\theta_2(\tau-\theta_2)\Delta t} \\ d_4 = \frac{\theta_2 - 1}{\tau(\theta_2-\tau)}, \quad d_5 = \frac{(\theta_2-1)(\tau-1)}{\tau\theta_2} \end{gathered} \tag{2.30}$$

For the second sub-step, we consider the equilibrium at $t = t_s + \Delta t$ as mentioned before, which can be stated as

$$\mathbf{M}\mathbf{a}_{s+1} + \mathbf{C}\mathbf{v}_{s+1} + \mathbf{K}_{s+1}\mathbf{u}_{s+1} = \mathbf{f}_{s+1} \tag{2.31}$$

By substituting Eqs. (2.29a) and (2.29b) into Eq. (2.31) we obtain

$${}^2\hat{\mathbf{K}}\mathbf{u}_{s+1} = {}^2\hat{\mathbf{f}} \tag{2.32}$$

where ${}^2\hat{\mathbf{K}}$ is the effective coefficient matrix of the second sub-step, $\mathbf{u}_{s+1}$ is the displacement vector at $t = t_s + \Delta t$, and ${}^2\hat{\mathbf{f}}$ is the effective force vector of the second sub-step. ${}^2\hat{\mathbf{K}}$ can be calculated as

$${}^2\hat{\mathbf{K}} = d_6\mathbf{M} + d_7\mathbf{C} + \mathbf{K}_{s+1} \tag{2.33}$$

where

$$d_6 = {d_1}^2, \quad d_7 = d_1 \tag{2.34}$$

and ${}^2\hat{\mathbf{f}}$ can be calculated as

$$\begin{aligned} {}^2\hat{\mathbf{f}} = \mathbf{M}\left(d_8\mathbf{u}_s + d_9\mathbf{u}_{s+\tau} + d_{10}\mathbf{v}_s + d_{11}\mathbf{v}_{s+\tau} + d_{12}\mathbf{a}_s + d_{13}\mathbf{a}_{s+\tau}\right) \\ + \mathbf{C}\left(d_{14}\mathbf{u}_s + d_{15}\mathbf{u}_{s+\tau} + d_{16}\mathbf{v}_s + d_{17}\mathbf{v}_{s+\tau}\right) + \mathbf{f}_{s+1} \end{aligned} \tag{2.35}$$

where

$$\begin{aligned} d_8 = -d_1 d_3, \quad d_9 = -d_1 d_2, \quad d_{10} = -(d_1 d_5 + d_3) \\ d_{11} = -(d_1 d_4 + d_2), \quad d_{12} = -d_5, \quad d_{13} = -d_4 \\ d_{14} = -d_3, \quad d_{15} = -d_2, \quad d_{16} = -d_5, \quad d_{17} = -d_4 \end{aligned} \tag{2.36}$$

Note that $\mathbf{u}_{s+\tau}$, $\mathbf{v}_{s+\tau}$ and $\mathbf{a}_{s+\tau}$ obtained from the first sub-step at each incremental time step.

2.2.4 Collocation Parameters and Dissipation

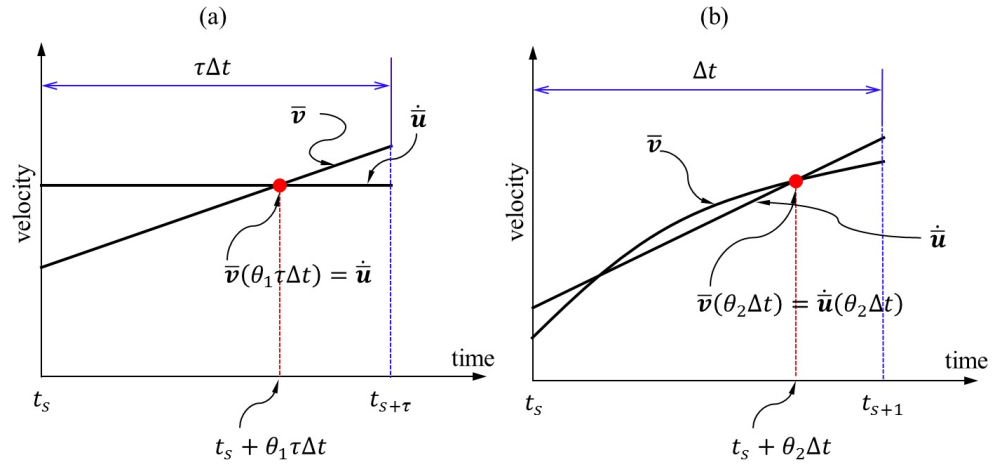

Figure 2.2: Schematic variation of approximated velocities: (a) the first sub-step; and (b) the second sub-step.

In the new algorithm, collocation parameters are important because they can adjust amount of algorithmic damping depending on locations within the time interval where the residuals are evaluated. Fig. 2.2 is showing how the residual equation given in Eq. (2.18a) is evaluated in the collocation sense. For the first sub-step, θ_1 plays a similar role of the weight effect parameter considered in Ref. [79] and alpha parameter in the generalized trapezoidal rule[6]. And the mid-point rule can be obtained by setting $\theta_1 = 0.5$, while backward scheme is obtained from $\theta_1 = 1.0$. If θ_1 is chosen as 0.5 in the first sub-step, then the evaluation of Eqs. (2.18a) and (2.18b) will be conducted at $t = t_s + 0.5\tau\Delta t$, which will give no-dissipation case. And choice of $\theta_1 = 1.0$ will give the asymptotic annihilation case. So the first sub-step will have various dissipation level depending on values of θ_1 chosen form 0.5 to 1.0.

For the second sub-step, θ_2 plays a similar role of θ_1 in the first sub-step. However, profile of $\dot{\bar{u}}$ and $\bar{v}$ are linear and quadratic respectively as shown in Fig. 2.2(b). Note that 3-point Euler backward method can be obtained with $\theta_2 = 1.0$. Thus choice of $\theta_1 = 0.5$, $\theta_2 = 1.0$, and $\tau = \gamma$ will give the Baig and Bathe method[35] where γ is the parameter used in the Baig and Bathe method. For this case, the complete algorithm becomes an asymptotic annihilation algorithm. Also the choice of $\theta_1 = 0.5$, $\theta_2 = 0.8535534$ and $\tau = 0.5$, will make the complete algorithm non-dissipative. Other combination of θ_1 and τ will change the specific value of θ_2 for specified levels of dissipation. We will relate two collocation parameters and the spectral radius in the high frequency limit (a user-specified dissipation control parameter) in the analysis section. Then a user will be able to specify any level of dissipation through the spectral radius in the high frequency limit.

2.3 Analysis

We can analyze the new time integration algorithm with a single -degree -of -freedom problem [49, 17, 88] of

$$\ddot{u} + 2\xi\omega\dot{u} + \omega^2 u = f \tag{2.37}$$

with the initial conditions

$$u(0) = u_0, \quad \dot{u}(0) = v_0 \tag{2.38}$$

For the free vibration case (i.e., $\xi = f = 0$) we can rewrite Eq. (2.37) by using the new algorithm given in Eqs. (2.21) and (2.32) as follows:

$$\mathbf{x}_{s+1} = \mathbf{A}\mathbf{x}_s, \quad s \in \{0, 1, ..., N-1\} \tag{2.39}$$

where $\{\mathbf{x}_i\} = \{u_i, v_i, a_i\}^T$ for $i = s,\ s+1$ and $\mathbf{A}$ is called the amplification matrix. We note that a_i can be condensed out by using the equilibriums at t_s and t_{s+1}. But, to keep the consistency with the original matrix and vector form of the discretized equation, we keep accelerations. Similarly, three by three amplification matrix was used in the work of Bathe and Noh[8] with the exactly the same equilibrium settings as ours. Then the computational characteristics of the new time integration algorithm can be studied by analyzing three eigenvalues of the amplification matrix (for details see Refs. [20, 49]). After rewriting the single-degree-of-freedom problem by using only the first sub-step, the spectral radius can be defined from the eigenvalues

of **A** by

$$\rho_1(\Omega) = \max\left(\left|\lambda_1^1\right|, \left|\lambda_2^1\right|, \left|\lambda_3^1\right|\right) \tag{2.40}$$

where λ_i^1 is the i^{th} eigenvalue of **A** obtained from the first sub-step scheme and $\Omega = \omega\Delta t$. One of three eigenvalues is always zero due to the equilibrium setting of current algorithm. Then the θ_1 can be selected according to

$$\theta_1 = \frac{1}{1 + \rho_1(\infty)} \tag{2.41}$$

where $\rho_1(\infty)$ is defined by

$$\rho_1(\infty) = \lim_{\Omega \to \infty} \rho_1(\Omega) \tag{2.42}$$

It should be noted that $0 \leq \rho_1(\infty) \leq 1.0$ does not violate the stability condition of the first sub-step. The stability condition of the first sub-step is given by

$$0.5 \leq \theta_1 \leq 1.0 \tag{2.43}$$

n a similar sense, we can also rewrite Eq. (2.37) in the form of Eq. (2.39) by combining the first and second sub-steps. Then θ_2 can be determined by conducting the same eigenvalues analysis of **A** constructed from the completely combined algorithm. The spectral radius of the new algorithm ρ_2 can be defined by

$$\rho_2(\Omega) = \max\left(\left|\lambda_1^2\right|, \left|\lambda_2^2\right|, \left|\lambda_3^2\right|\right) \tag{2.44}$$

where λ_i^2 is the i^{th} eigenvalue of **A** obtained from the new algorithm which combines

Eqs. (2.21) and (2.32). Here we define spectral radius of the new algorithm as

$$\rho_2(\infty) = \lim_{\Omega\to\infty} \rho_2(\Omega) \tag{2.45}$$

and $\rho_2(\infty)$ determines the dissipation of the combined algorithm at high frequency limit. Since $\rho_2(\infty)$ determines the dissipation of the combined complete algorithm, we may regard $\rho_2(\infty)$ as the spectral radius in high frequency limit (i.e., ρ_∞) of the complete algorithm. In current study $\rho_2(\infty)$ and ρ_∞ are interchangeably used, and θ_2 is determined by

$$\begin{aligned}\theta_2 &= \frac{\tau^2\theta_1(\rho_\infty - 1) + 1}{2(1 - \tau\theta_1(1 - \rho_\infty))} \\ &+ \frac{\sqrt{\tau^4{\theta_1}^2(\rho_\infty - 1)^2 - 4\tau^2(1-\tau){\theta_1}^2(\rho_\infty - 1) + 2\tau^2\theta_1(\rho_\infty + 1) - 4\tau\theta_1 + 1}}{2(1 - \tau\theta_1(1 - \rho_\infty))}\end{aligned} \tag{2.46}$$

where θ_1 is usually selected as 0.5 which can be determined from Eq. (2.41) by setting $\rho_1(\infty) = 1.0$. It should be emphasized that the new algorithm can work as the no dissipation case ($\rho_1(\infty) = 1.0$, $\rho_2(\infty) = 1.0$) and include the Baig and Bathe method ($\rho_1(\infty) = 1.0$, $\rho_2(\infty) = 0.0$) as special cases.

In the new algorithm, the second sub-step will control the high frequency limit dissipation level of the complete algorithm, which is very similar to the mechanism of the Baig and Bathe method. In fact the complete algorithm can remain as no-dissipation case ($\rho_\infty = 1.0$), even if $\rho_1(\infty) = 0.0$. However the accuracy will be affected. In the Baig and Bathe method the second sub-step works as the asymptotic annihilation case ($\rho_\infty = 0.0$), while the first sub-steps cannot provide any algorithmic damping because it is the trapezoidal rule. Unlike the Baig and Bathe method, the new algorithm can control the dissipation level of the first sub-step through θ_1 (by specifying $\rho_1(\infty)$) and proper choice of θ_1 can contribute to the stability and increase

the quality of numerical solutions in some severe nonlinear problems. However, we limit our study only to the case of $\theta_1 = 0.5$ (i.e., $\rho_1(\infty) = 1.0$) to retain the second order accuracy of the complete algorithm. If θ_1 is selected from the interval of $0.5 < \theta_1 \leq 1.0$ instead of 0.5, the complete algorithm loses the second-order accuracy and becomes first-order accurate.

2.3.1 Stability

The new algorithm is unconditionally stable [49, 25, 89] if $0.0 \leq \rho_1(\Omega) \leq 1.0$ and $0.0 \leq \rho_2(\Omega) \leq 1.0$ are provided for all $\Omega \in (0, \infty)$. It can be easily shown that the new algorithm is stable for any combination of θ_1 and θ_2 which are determined by Eqs. (2.41) and (2.46).

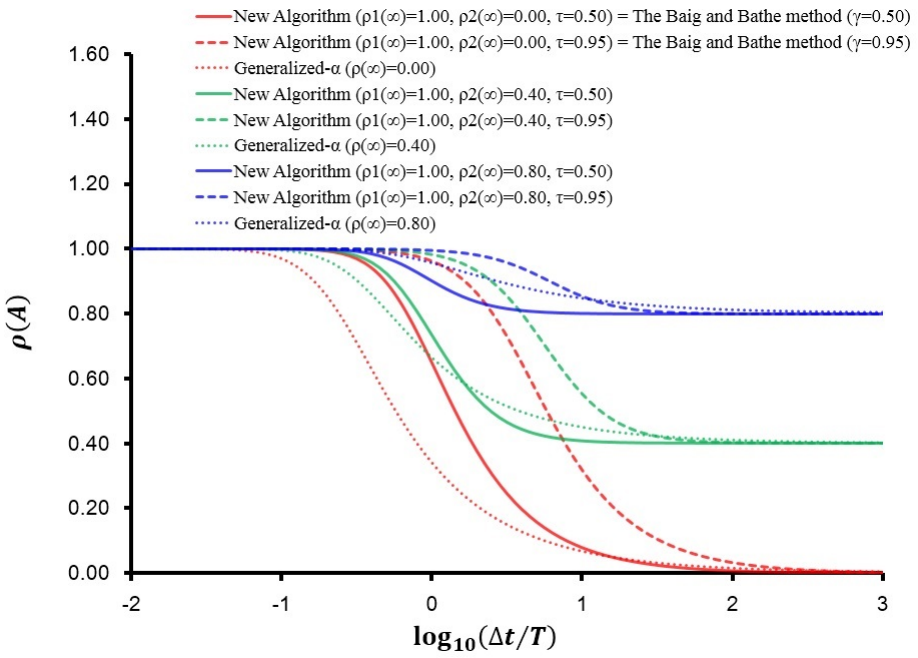

Figure 2.3: Comparison of spectral radii

Fig. 2.3 shows that the new algorithm is unconditionally stable and more accurate for the large dissipative cases ($\rho_\infty = 0.0$ and 0.4) compared with the generalized-α method. Usually practical analyses choose Δt in the range of $\Delta t \leq T/10$, where T is period of the given dynamic system and Δt is chosen size of the time step. Having $\rho \simeq 1.0$ around $\Delta t/T = 0.1$ is considered as one of the desirable properties of an effective time integration algorithm, because an algorithm should not become too much dissipative for this choice of Δt. The spectral radius of the new algorithm is closer to unity (=0.9995) when $\tau = 0.5$ while the generalized-α method has noticeably decreased value (=0.9697) from the unity at $\Delta t/T = 0.1$ when $\rho_\infty = 0.0$. However, the generalized-α method shows slightly better characteristics of the spectral radius than the new algorithm for $\rho_\infty > 0.5$ when $\tau = 0.5$ is used for the new algorithm. But the spectral radius of the new algorithm can be improved by adjusting τ as presented in Fig. 2.5. But the choice of τ that is too close to 1.0 should be avoided. Here we only presented the effects of some values of τ on the spectral radius of the algorithm. The effects of τ on the cases of $\rho_\infty = 0.0$ and $\rho_\infty = 0.8$ are presented in Figs. 2.4 and 2.5, respectively. Details of τ will be discussed in sequel.

2.3.2 Accuracy

In many cases, the order of accuracy is defined from the local truncation error[20, 49, 17]. The local truncation error τ_e is defined by

$$\tau_e = (\Delta t)^{-2} \sum_{3}^{i=0} (-1)^i A_i u(t_{n+1-i}) \tag{2.47}$$

where $A_0 = 1$, A_1 is the trace of $\mathbf{A}$, A_2 is the sum of principal minors of $\mathbf{A}$, A_3 is the determinant of $\mathbf{A}$ and $u(t)$ is the exact solution of the problems. Then the algorithm is called the kth order algorithm if $\tau_e = O(\Delta t^k)$ is provided. For any setting of

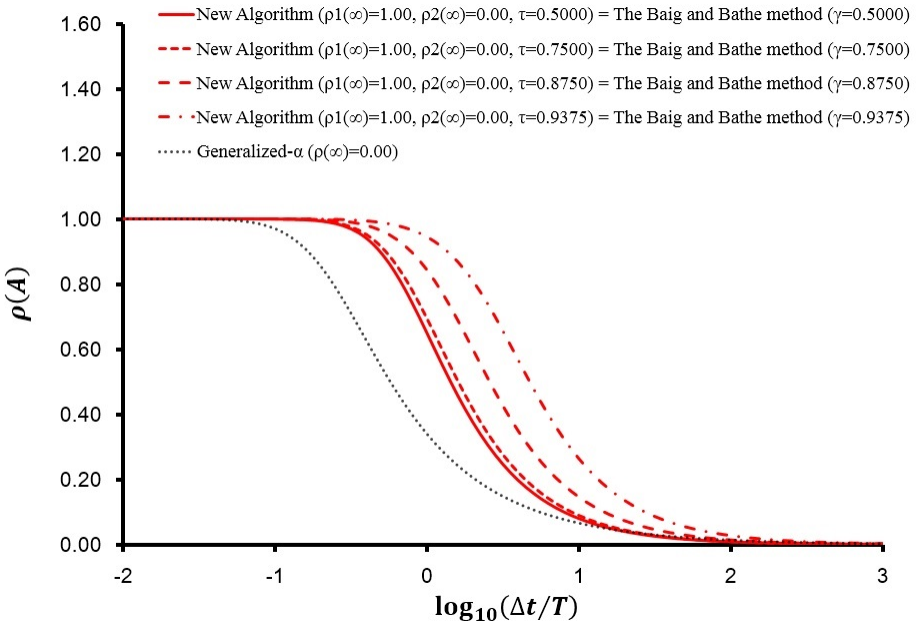

Figure 2.4: Effect of τ on spectral radius of new algorithm for $\rho_\infty = 0.0$

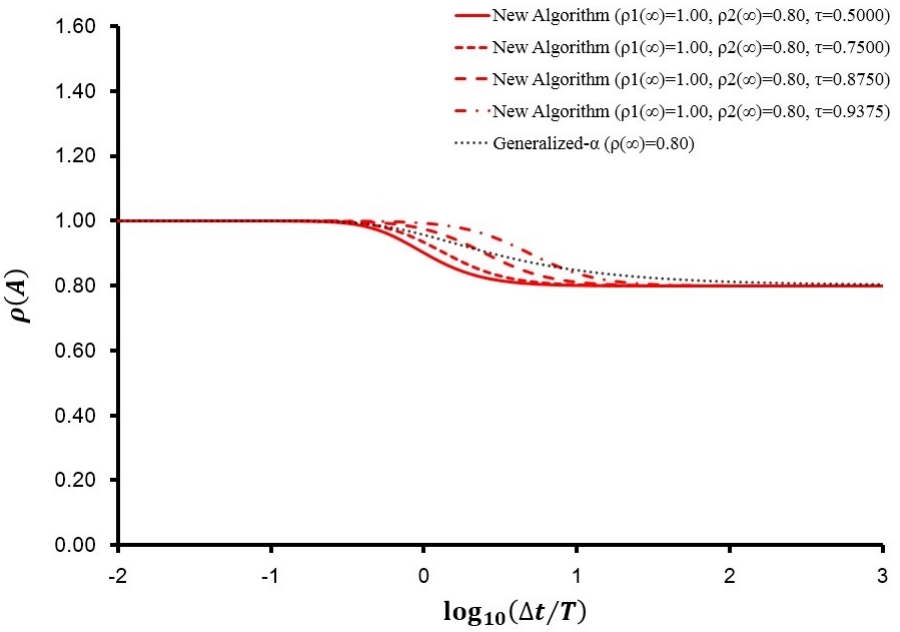

Figure 2.5: Effect of τ on spectral radius of new algorithm for $\rho_\infty = 0.8$

admissible parameters with $\theta_1 = 0.5$, the new algorithm provides $\tau_e = O(\Delta t^2)$, thus the new algorithm is second-order accurate if $\theta_1 = 0.5$.

In the literature[27, 7], the accuracy of the time integration algorithms is also explained by using the period elongation and the damping ratio. Both the period elongation and the damping ratio are properties which are obtained by comparing the algorithmic solution of Eq. (2.37) with the exact solution. In this paper, $\bar{T}$ and $\bar{\xi}$ are used for the period and damping of the algorithmic solution. The comparison of the period elongation and the damping ratio can provide more direct information about the algorithm than the accuracy rate (i.e., the convergence rate) which is still important mathematically. The period elongation and the damping ratio of the new algorithm are compared with those of the generalized-α method in Figs. 2.6 and 2.7.

In the generalized-α method, period elongation and damping ratio are noticeably large for $\Delta t \geq T/10$ for large dissipative cases (i.e., $0.0 \leq \rho_\infty \leq 0.5$). Especially, the asymptotic annihilation case (i.e., $\rho_\infty = 0.0$) of the generalized-α method has noticeably large period elongation as shown in Figs. 2.8 and 2.10. For the new algorithm case of $\rho_1(\infty) = 1.0$, $\rho_\infty = 0.0$ and $\tau = 0.5$ is identical to the Baig and Bathe method which is already proven to be very effective asymptotic annihilation algorithm. For some values of τ which is chosen from $0.5 \leq \tau < 1.0$, the damping ratio can be even improved while the period elongation increases slightly. Both the generalized-α method and the new algorithm seem to perform nicely for $\rho_\infty = 0.8$ as presented in Figs. 2.9 and 2.11. However, the new algorithm can have even a better damping ratio by adjusting τ as shown in Fig. 2.11 when $\rho_\infty > 0.5$ by slightly sacrificing the period elongation as shown in Fig. 2.9.

2.3.3 General Comments on the Choice and Effect of τ

In the new algorithm τ is included as one of three free parameters. It is shown that τ can affect the spectral radius, damping ratio and period elongation in previous sections. Here we explain some important features of τ.

First, for τ selected from $0.5 \leq \tau < 1.0$, the stability of the linear system will not be affected. But in nonlinear analyses, τ which is too close to 1.0 should be avoided. If τ is too close to 1.0, noticeable decrease of the spectral radius will start at relatively higher frequency range as shown in Fig. 2.4 and 2.5. Then less algorithmic damping will be introduced in low frequency ranges. If the algorithm fail to bring out enough amount of algorithmic damping into low frequency ranges with large time interval, system can become unstable like in the case of the trapezoidal rule. However, choice of τ from $0.5 \leq \tau \leq 0.90$ is considered safe according to our experience.

Second, proper selection of τ can improve the damping ratio while slightly decreasing the period accuracy. Our numerical results report that the period elongation and the damping ratio are in trade-off type relation. As τ approaches to 1.0 (but it should be less than 1.0), we obtain less algorithmic damping while the period error increases. Thus user should be aware of this for a proper selection of τ, otherwise we recommend $\tau = 0.5$ as a standard case.

Third, proper choice of τ can be used to increase the computational efficiency in linear analysis. We note that τ in our algorithm plays a similar role of γ in the Baig and Bathe method[35, 40] even though effect of γ on accuracy of the algorithm was not explained in detail in their original works. Recently, some roles of γ in the Baig and Bathe method was discussed in Ref. [90]. In Ref. [41], a single effective coefficient matrix was considered for the first and second sub-steps by setting $\gamma = 2 - \sqrt{2}$. If we chose $\rho_1(\infty) = 1.0$, $\rho_2(\infty) = 0.0$, and $\tau = 2 - \sqrt{2}$ in our algorithm, we obtained the

same single effective coefficient matrix as that of the Baig and Bathe method. But main difference between τ in our algorithm and γ in the Baig and Bathe method is that τ works for every range of spectral radius while use of γ is fixed for the asymptotic case only. For example, choices of $\rho_1(\infty) = 1.0$, $\rho_2(\infty) = 0.5$, and $\tau = 4 - 2\sqrt{3} \approx 0.5358983848$ in our algorithm will gives another single effective coefficient matrix. In linear case (with constant $\mathbf{K}$), this single effective coefficient matrix is computed as

$$^1\hat{\mathbf{K}} =^2 \hat{\mathbf{K}} = \frac{13.928203}{\Delta t^2}\mathbf{M} + \frac{3.7320508}{\Delta t}\mathbf{C} + \mathbf{K} \tag{2.48}$$

And use of Eq. (2.48) will require only single diagonalization of effective coefficient matrix which is a huge saving in linear case as mentioned in [41].

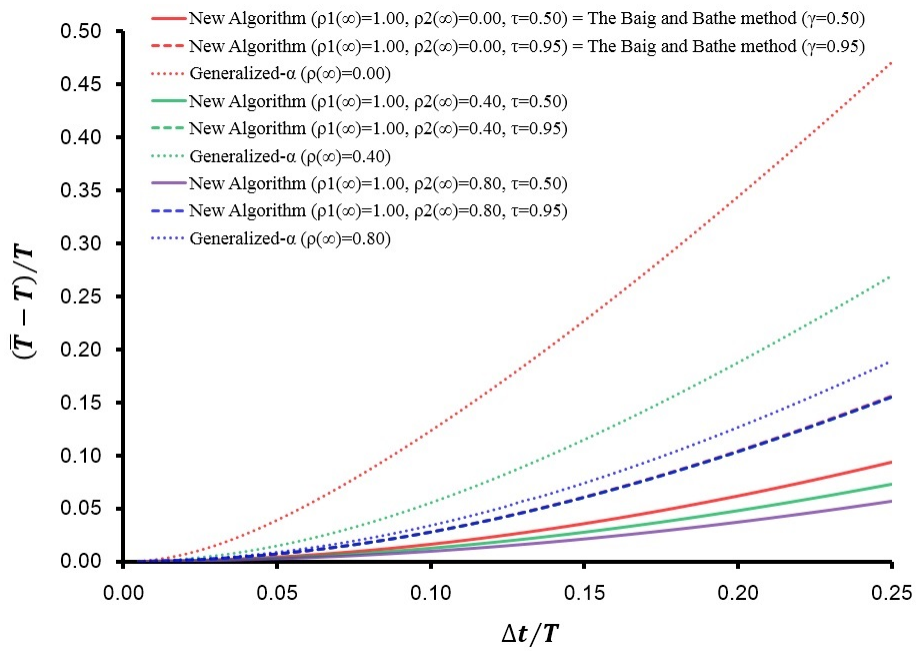

Figure 2.6: Comparison of period elongations

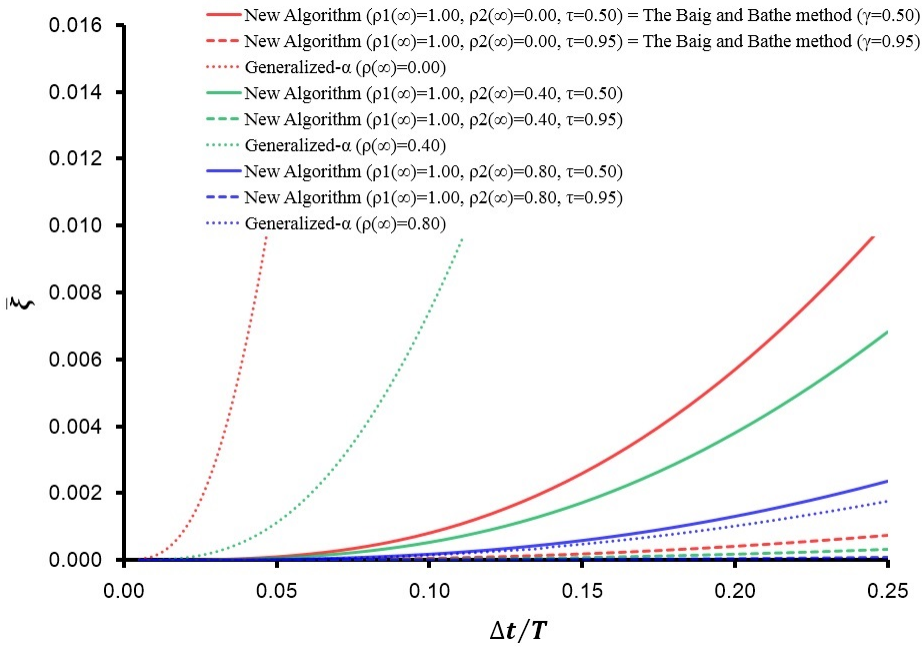

Figure 2.7: Comparison of damping ratios

2.4 Numerical Examples

We have analyzed the new algorithm using the single-degree-of-freedom problem. Here we verify its performance with several multi-degree-of-freedom problems. First we begin with linear problems. We analyze a linear multi-degree of freedom spring problem proposed by Bathe and Noh [8]. In the original problem of Bathe and Noh, the high frequency filtering capability of the Baig and Bathe method was demonstrated with this problem by using very stiff and flexible spring constants. But in our study, we modify this problem to demonstrate some potential misuse of general asymptotic annihilation algorithms. Then a linear 2-D standing wave type problem [91] is solved by using algorithms, and the analytical series solution is used for a precise comparison. As a nonlinear problem, the nonlinear FSDT (first-order

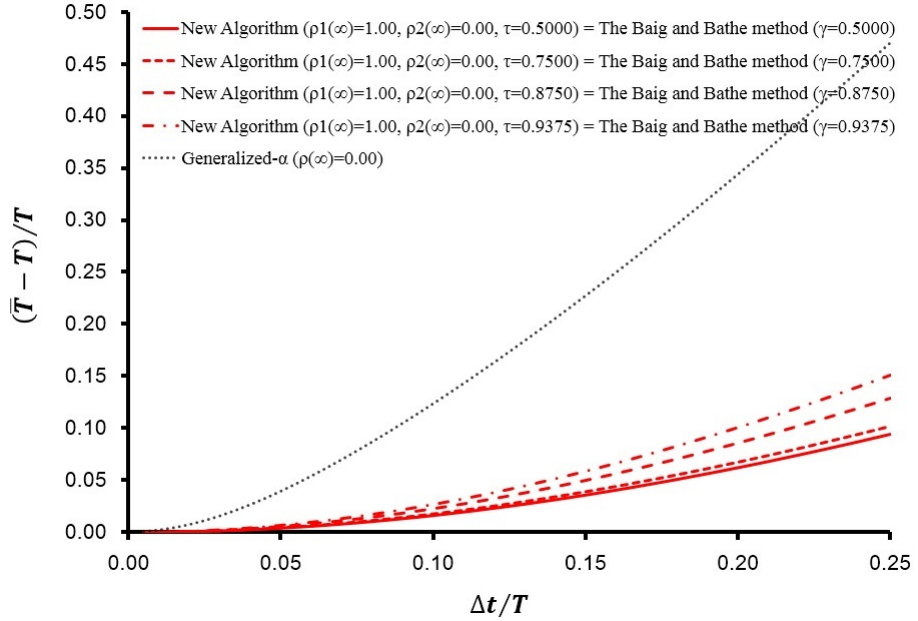

Figure 2.8: Effect of τ on period elongation of new algorithm for $\rho_\infty = 0.0$

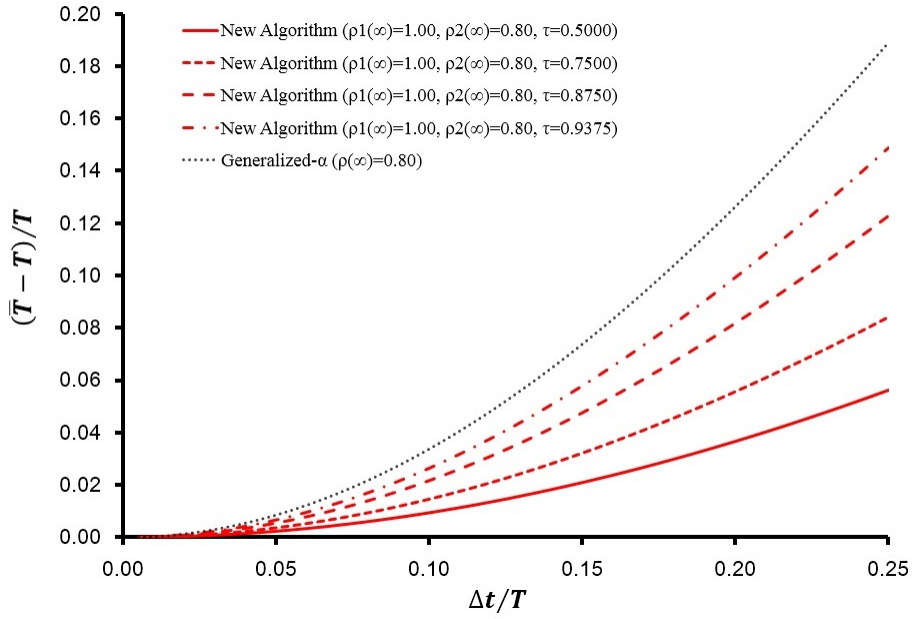

Figure 2.9: Effect of τ on period elongation of new algorithm for $\rho_\infty = 0.8$

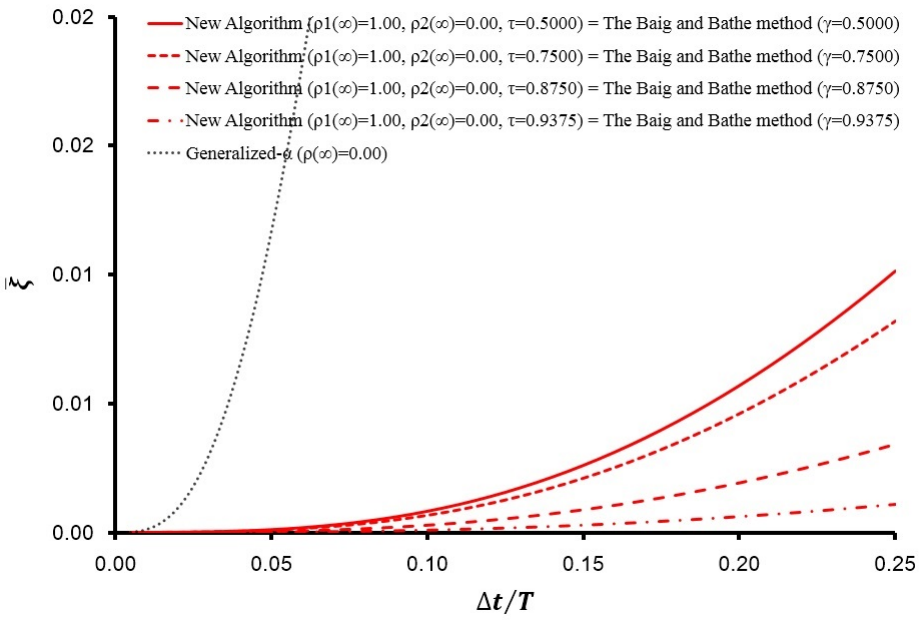

Figure 2.10: Effect of τ on damping ratio of new algorithm for $\rho_\infty = 0.0$

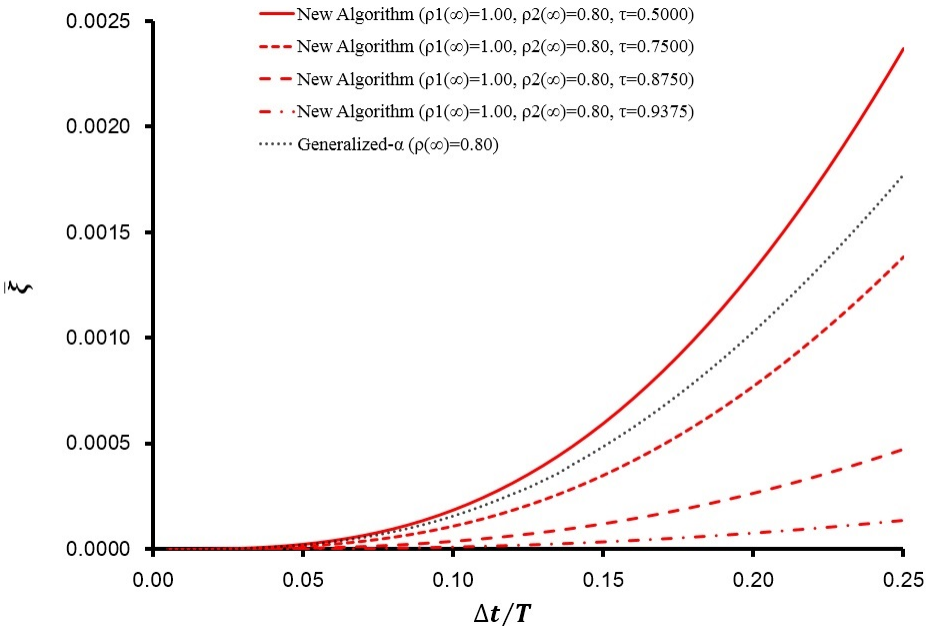

Figure 2.11: Effect of τ on damping ratio of new algorithm for $\rho_\infty = 0.8$

shear deformation theory) plate problem given in Ref. [4] is analyzed with the new algorithm. Numerical results are compared with the results obtained from various schemes.

2.4.1 Linear Problems

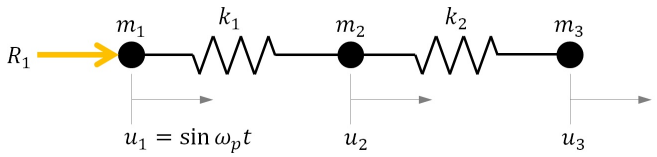

Figure 2.12: Description of three degrees of freedom spring system used by Bathe and Nho.

Bathe and Noh used a three degrees of freedom spring system shown in Fig. 2.12 to represent a simplified version of the complex structural system, which consists of stiff and flexible parts. But we will use the same problem to represent rather general simplified structural system with a moderate difference in stiffness. The governing equation of the spring system shown in Fig. 2.12 is given by

$$\begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix} \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \end{Bmatrix} + \begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1 + k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} R_1 \\ 0 \\ 0 \end{Bmatrix} \tag{2.49}$$

where $u_1 = \sin \omega_p t$ and R_1 is the reaction force at node 1. The initial conditions of the zero displacement, velocity, and acceleration vectors (i.e., $u_i = 0$, $v_i = 0$ and $a_i = 0$ for $i = 1, 2, 3$) are used. Then by using the prescribed displacement at the

first node, Eq. (2.49) can be reduced to

$$\begin{bmatrix} m_2 & 0 \\ 0 & m_3 \end{bmatrix} \begin{Bmatrix} a_2 \\ a_3 \end{Bmatrix} + \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix} \begin{Bmatrix} u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} k_1 u_1 \\ 0 \end{Bmatrix} \tag{2.50}$$

Now we solve Eq. (2.50) by using the new algorithm, the generalized-α method and the trapezoidal rule. A test problem of completely different characteristic can be generated by simply adjusting one of spring constants of the original problem, and it can be used to demonstrate potential misuse of the asymptotic annihilation case and advantage of proper dissipation control. We use $k_1 = 50$, $k_2 = 1$, $m_1 = 0$, $m_2 = 1$, $m_3 = 1$, and $\omega_p = 1.0$. Only k_1 has been significantly modified from the original problem. With these specific problem data, the period of the prescribed displacement becomes $T_p = 6.28185$ and the two natural periods of the system become $T_1 = 6.346949$ and $T_2 = 0.8796495$. Since the ratio T_1 to T_2 is just 7.215, we assume that both frequency modes are important and their effect should be included in numerical solutions. To make our comparison even clearer, we use the exact modal superposition solution of the modified problems for the comparison. The exact solution of the second and third nodes are obtained by

$$\begin{aligned} u_2 =& 0.1398887047\ \sin(7.142828012\ t) + 1.009341274\ \sin(0.9899535310\ t) \\ u_3 =& -50.00\ \sin(t) + 0.0027966559\ \sin(7.142828012\ t) \\ & + 50.48724248\ \sin(0.9899535310\ t) \end{aligned} \tag{2.51}$$

In this problem, displacement solutions does not present any noticeable differences between numerical solutions and the exact solution as shown in Fig. 2.13. As Bathe and Nho[8] did, we use T_p to select Δt. We use $\Delta t = T_p/10 = 0.6283185$, $\Delta t = T_p/20 = 0.3141592$ and $\Delta t = T_p/40 = 0.1570796$ for our analysis. For For

$\Delta t = T_p/10$ and $\Delta t = T_p/20$, numerical solutions of the velocity and acceleration obtained from the no-dissipation cases (the trapezoidal rule and the new algorithm with $\rho_1(\infty) = 1.0$, $\rho_2(\infty) = 1.0$ and $\tau = 0.5$) are very similar (but with noticeable period errors) to the exact solution as can be seen in Figs. 2.14 and 2.15 and Figs. 2.17 and 2.18. However the asymptotic annihilation cases ($\rho_1(\infty) = 1.0$, $\rho_2(\infty) = 0.0$, the Baig and Bathe method and the generalized-α method with $\rho_\infty = 0.0$) are providing very distorted velocity and acceleration solutions at the second node. If we suppose that the problem in hand is much more complicated and nonlinear, user may not know this kind of elimination of important frequencies. Thus less dissipative cases including the no-dissipations case can be used together to prevent unnotified elimination of important frequency modes. It should be also noted that better numerical solutions can be obtained with asymptotic annihilation cases with smaller $\Delta t (= 0.1570796)$ as shown in Figs. 2.20 and 2.21, but after relatively long time, we found noticeable amplitude decay. Of course, all numerical solutions can be improved by using even smaller Δt. Then the computational cost will also increase and the advantage of the implicit method over the explicit method will be lost.

The specific computational procedure of the linear analysis is provided in Table 2.1 and 2.2.

2.4.2 Nonlinear Problems

As a nonlinear test problem we use the bending of an isotropic plate using the FSDT. The specific computational procedure of the nonlinear analysis is provided in Tables 1 and 3.

The governing equations of the FSDT [4, 1] are given as

$$\rho h \frac{\partial^2 u_0}{\partial t^2} - \frac{\partial N_{xx}}{\partial x} - \frac{\partial N_{xy}}{\partial y} = 0 \tag{2.52a}$$

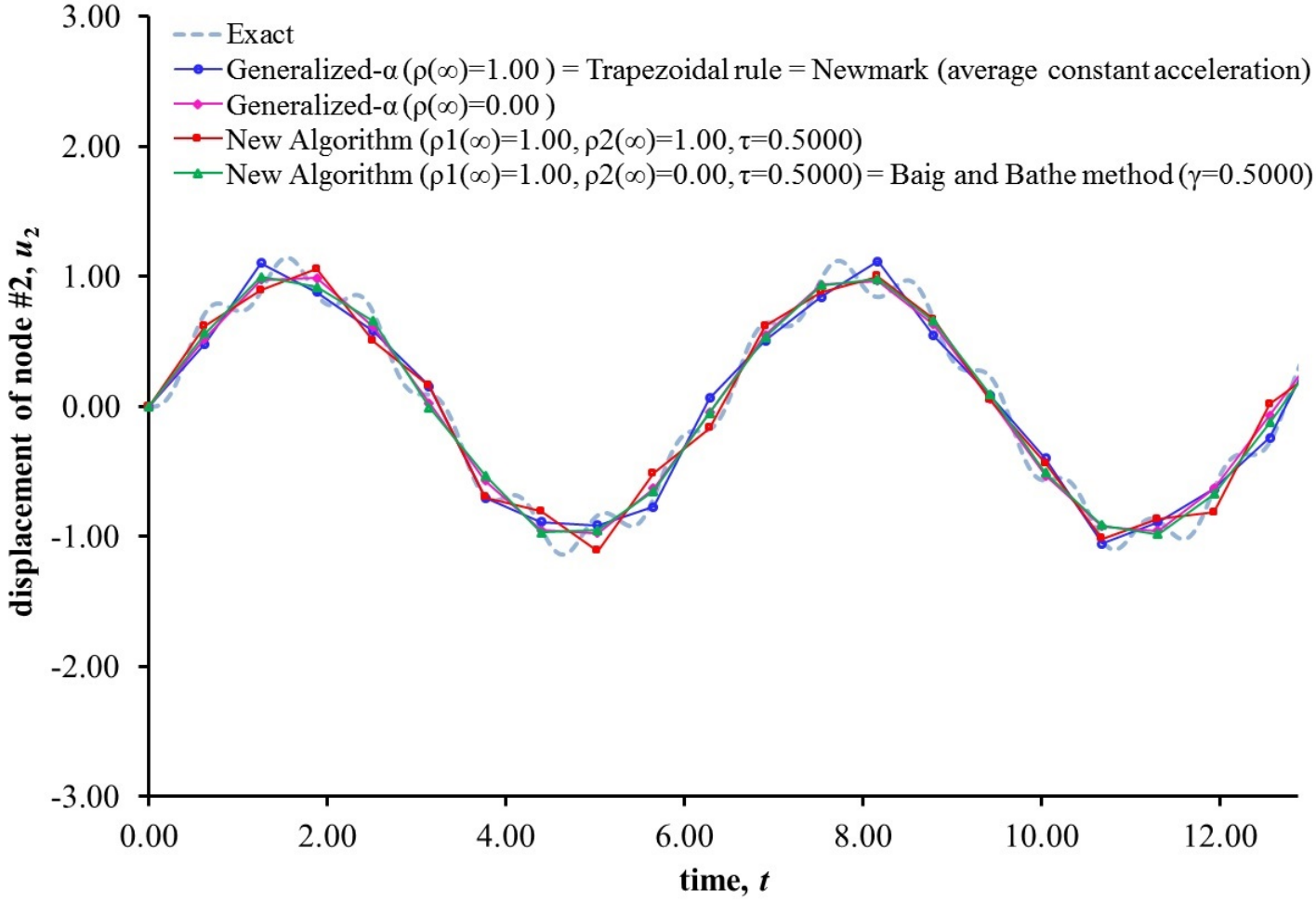

Figure 2.13: Displacements of node 2 (u_2) with $\Delta t = T_p/10$

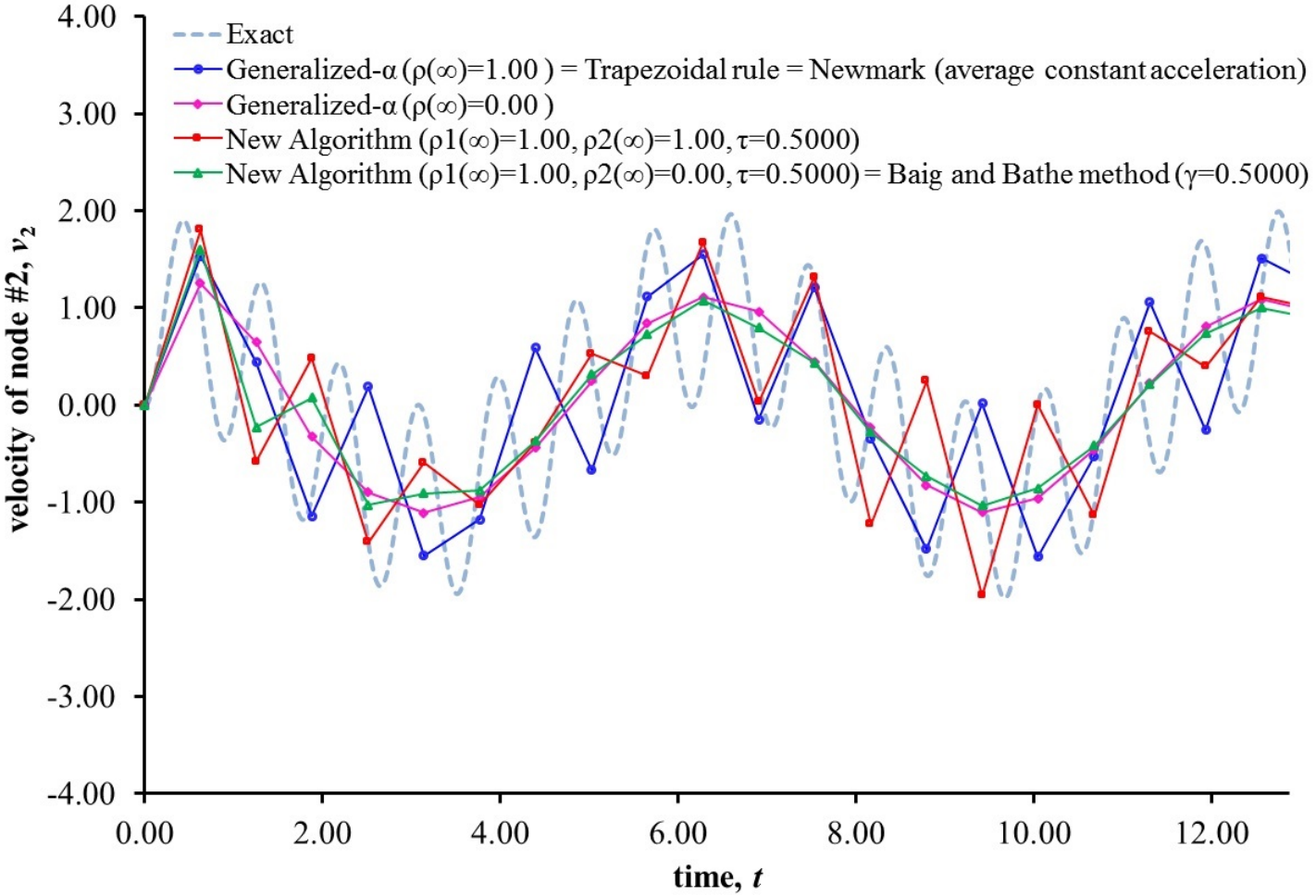

Figure 2.14: Velocities of node 2 (v_2) with $\Delta t = T_p/10$

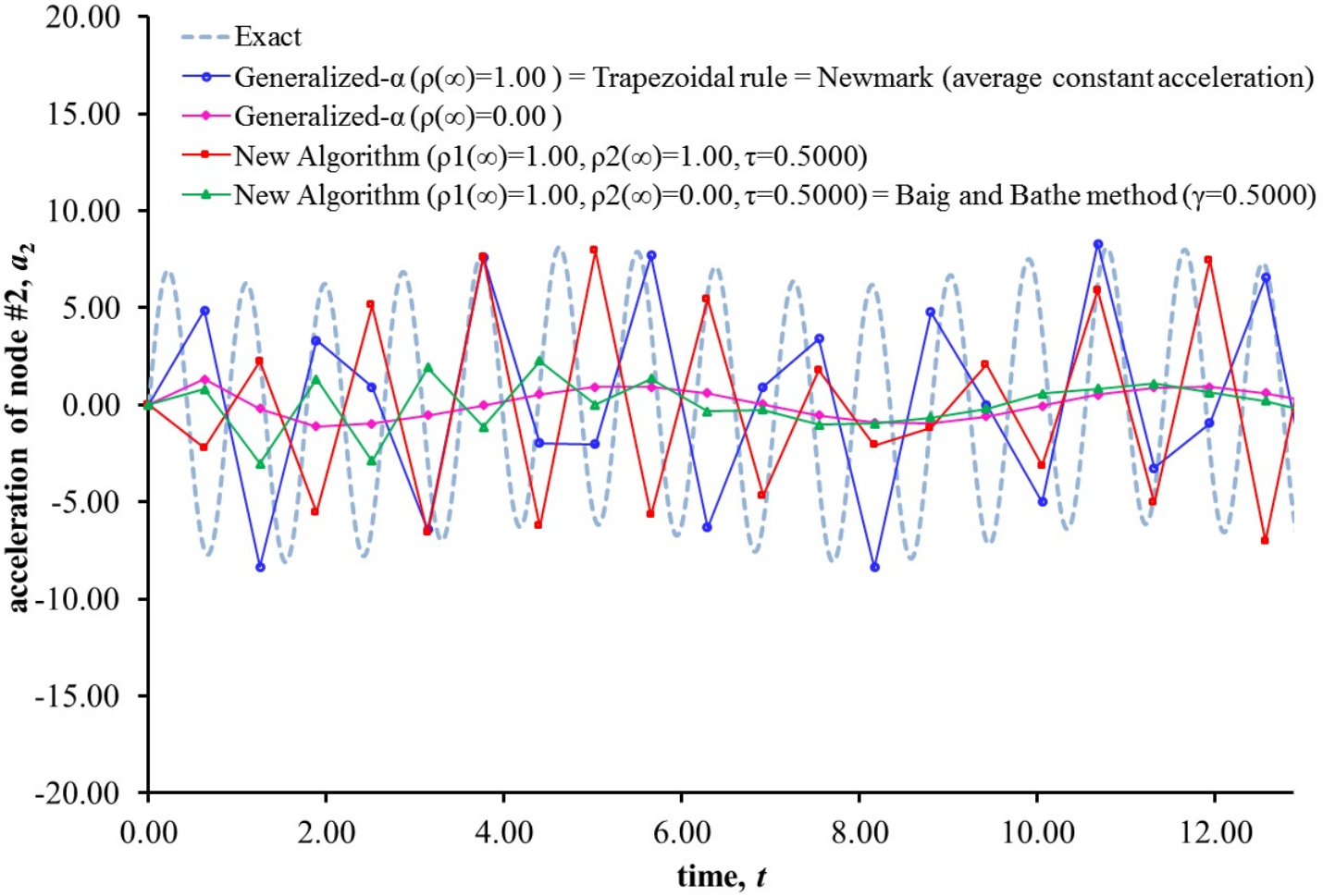

Figure 2.15: Accelerations of node 2 (a_2) with $\Delta t = T_p/10$

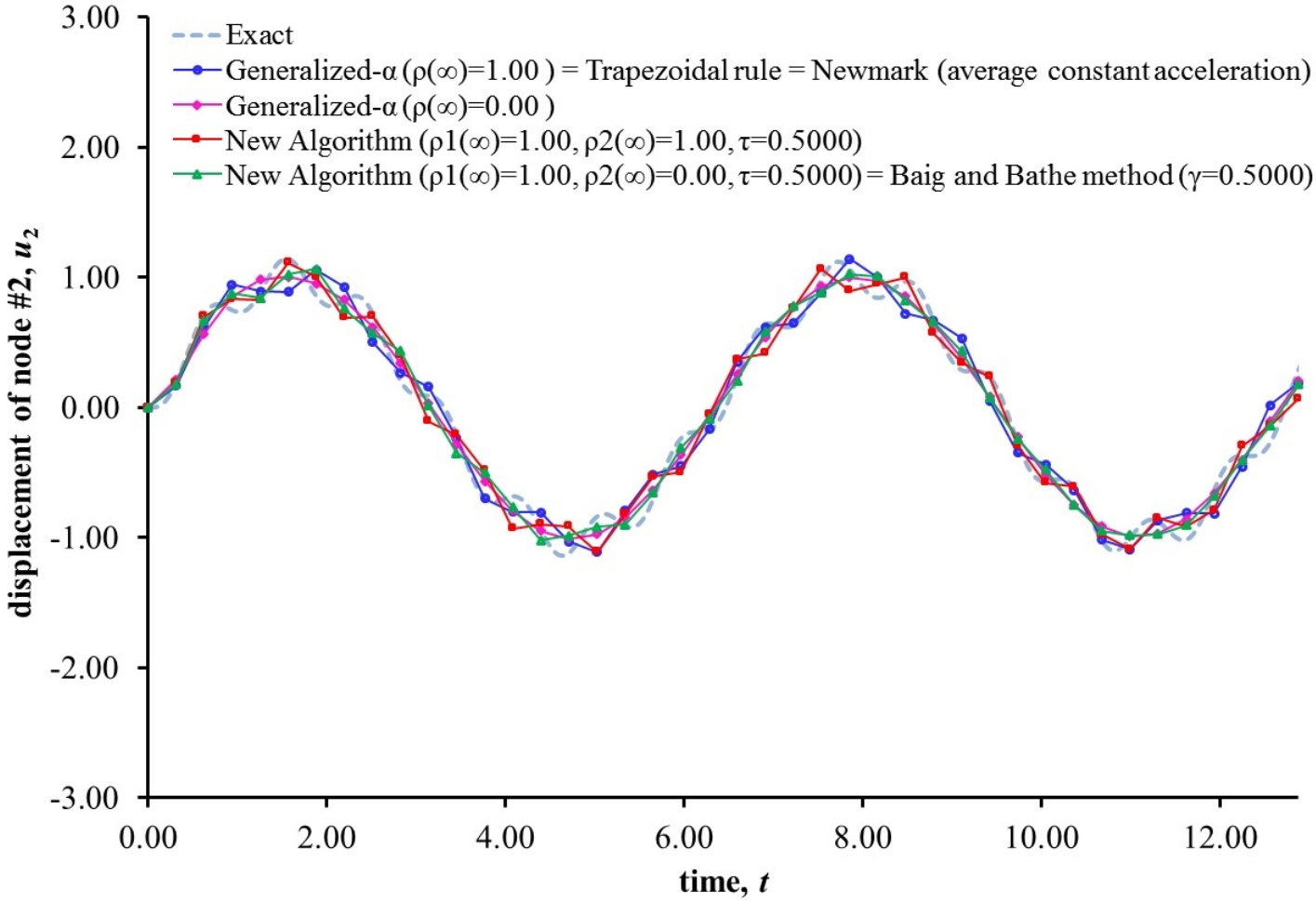

Figure 2.16: Displacements of node 2 (u_2) with $\Delta t = T_p/20$

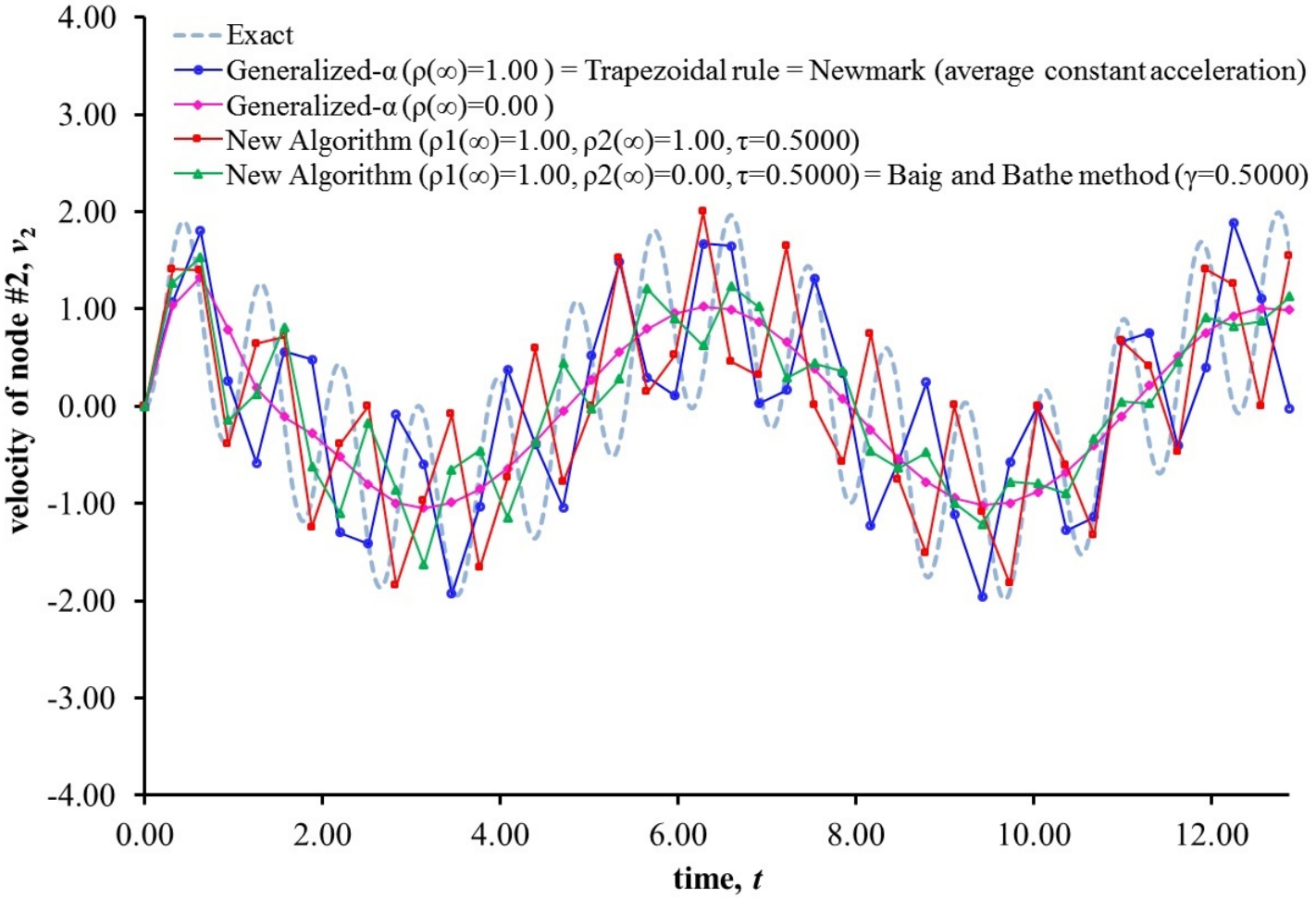

Figure 2.17: Velocities of node 2 (v_2) with $\Delta t = T_p/20$

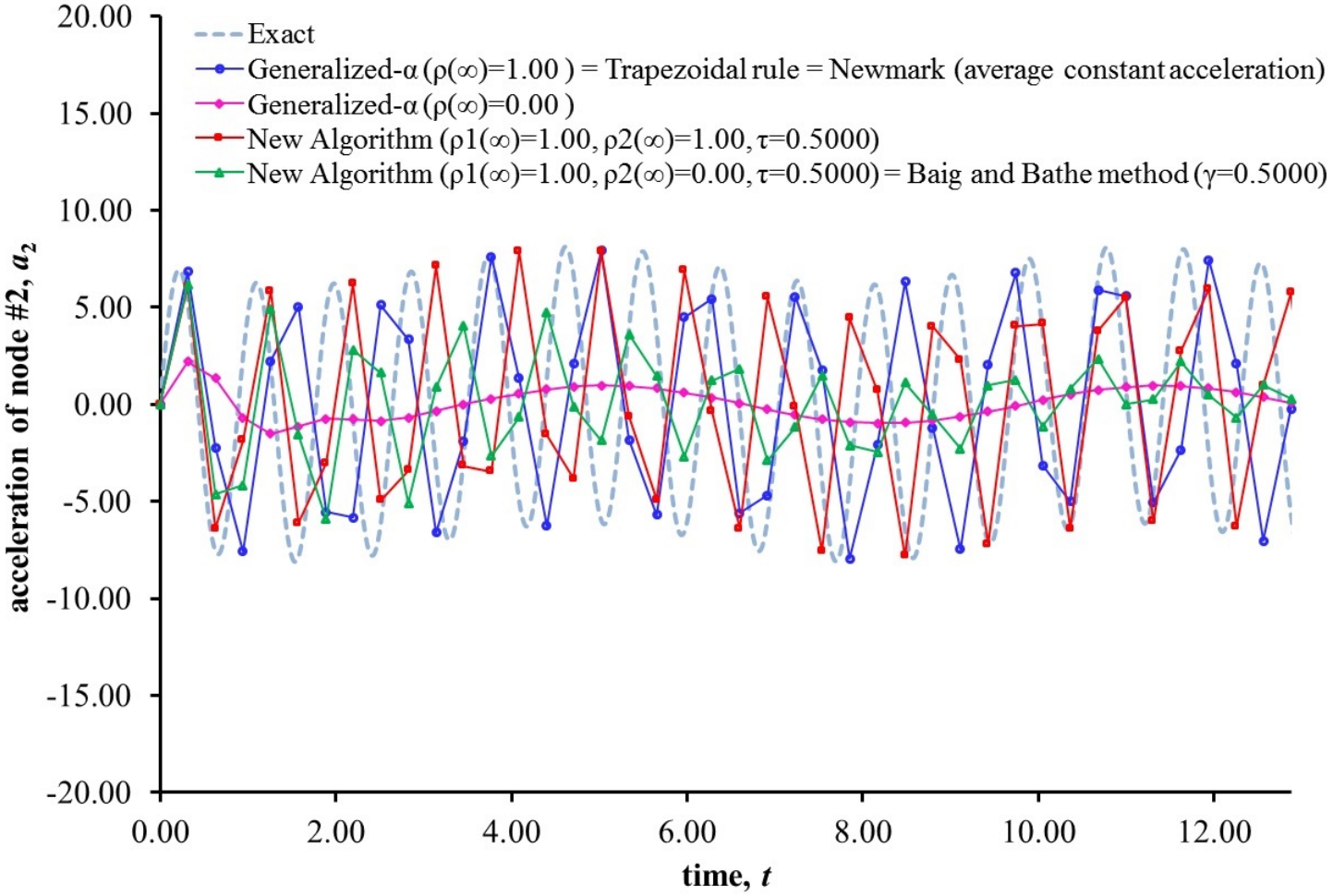

Figure 2.18: Accelerations of node 2 (a_2) with $\Delta t = T_p/20$

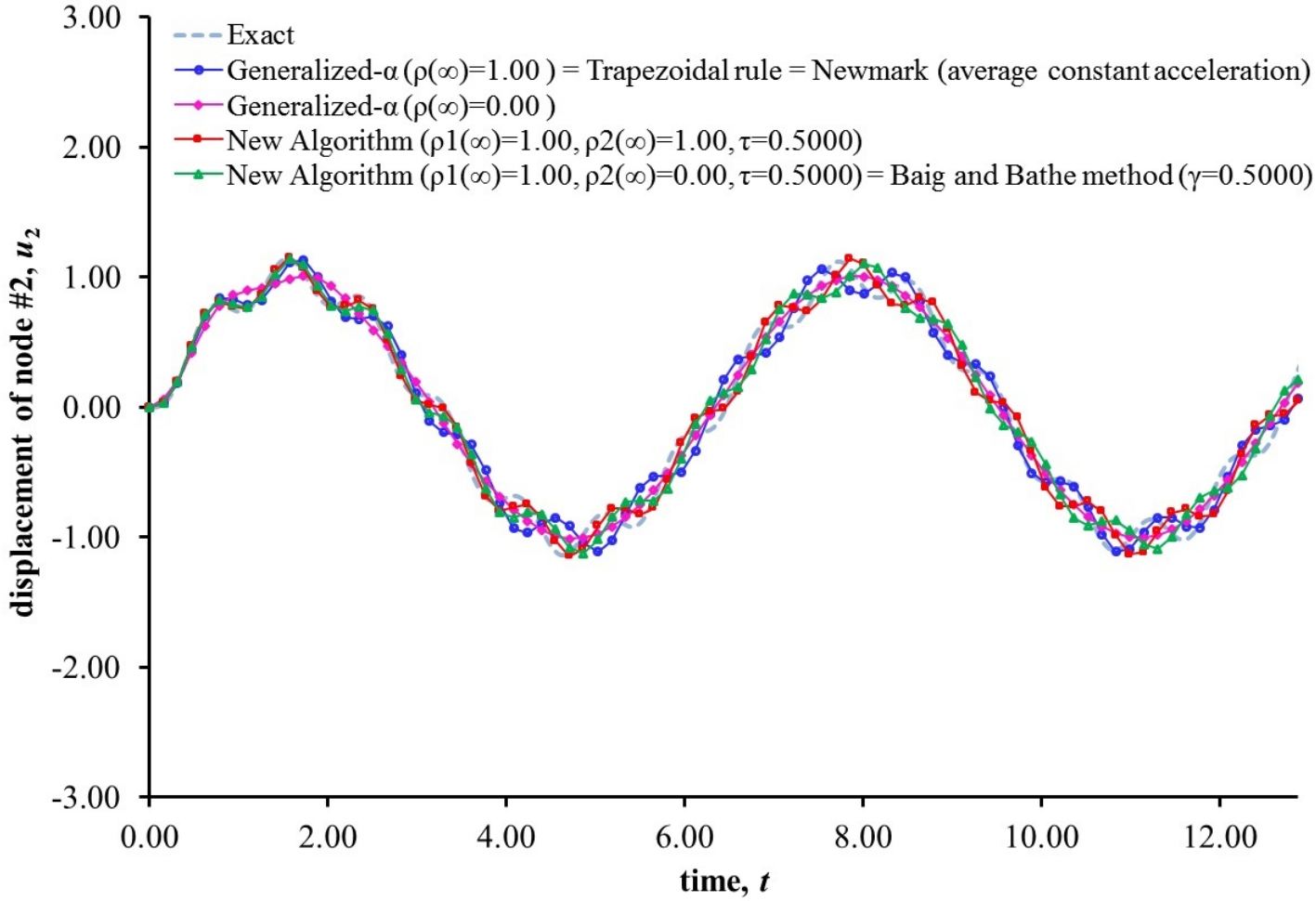

Figure 2.19: Displacements of node 2 (u_2) with $\Delta t = T_p/40$

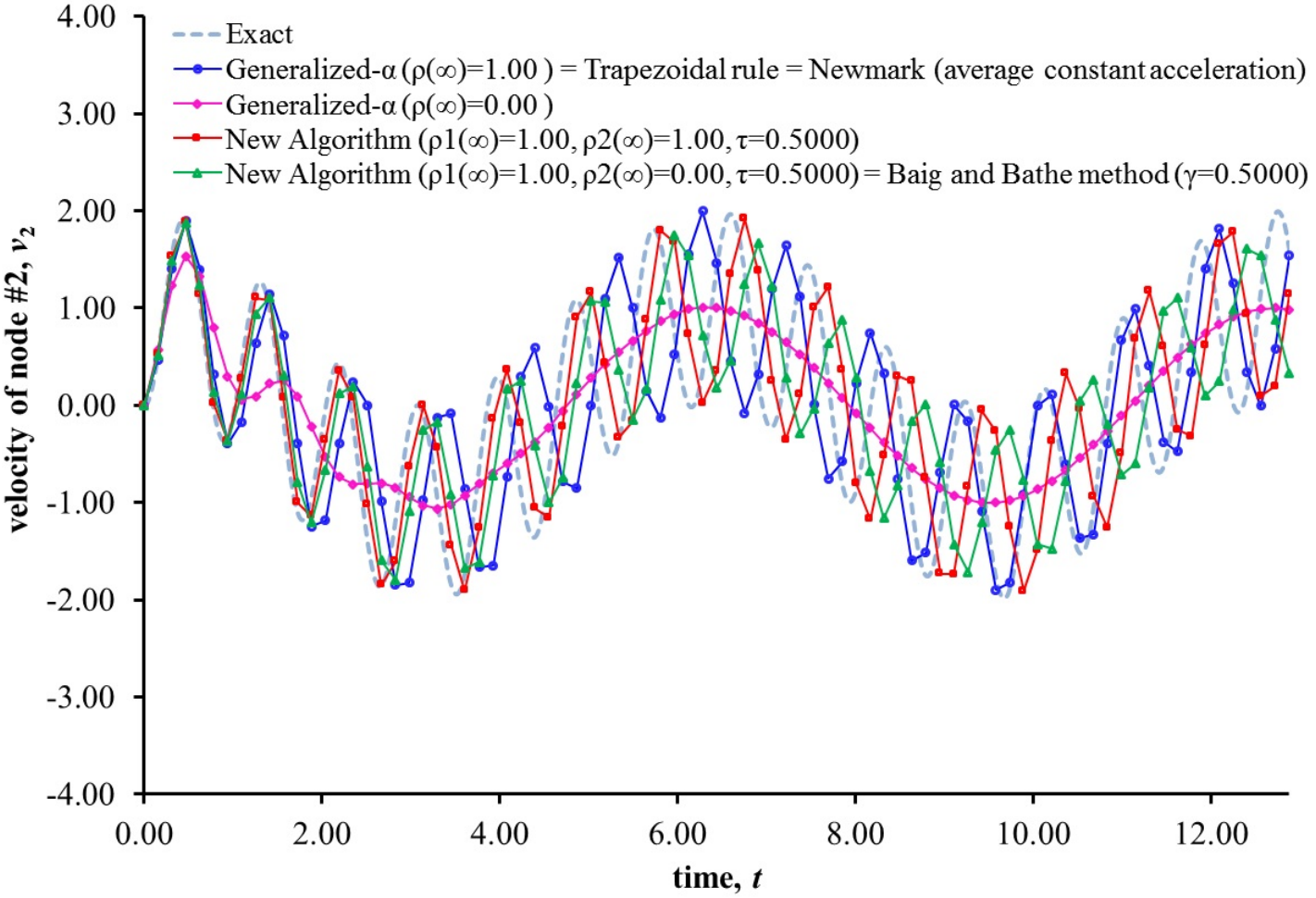

Figure 2.20: Velocities of node 2 (v_2) with $\Delta t = T_p/40$

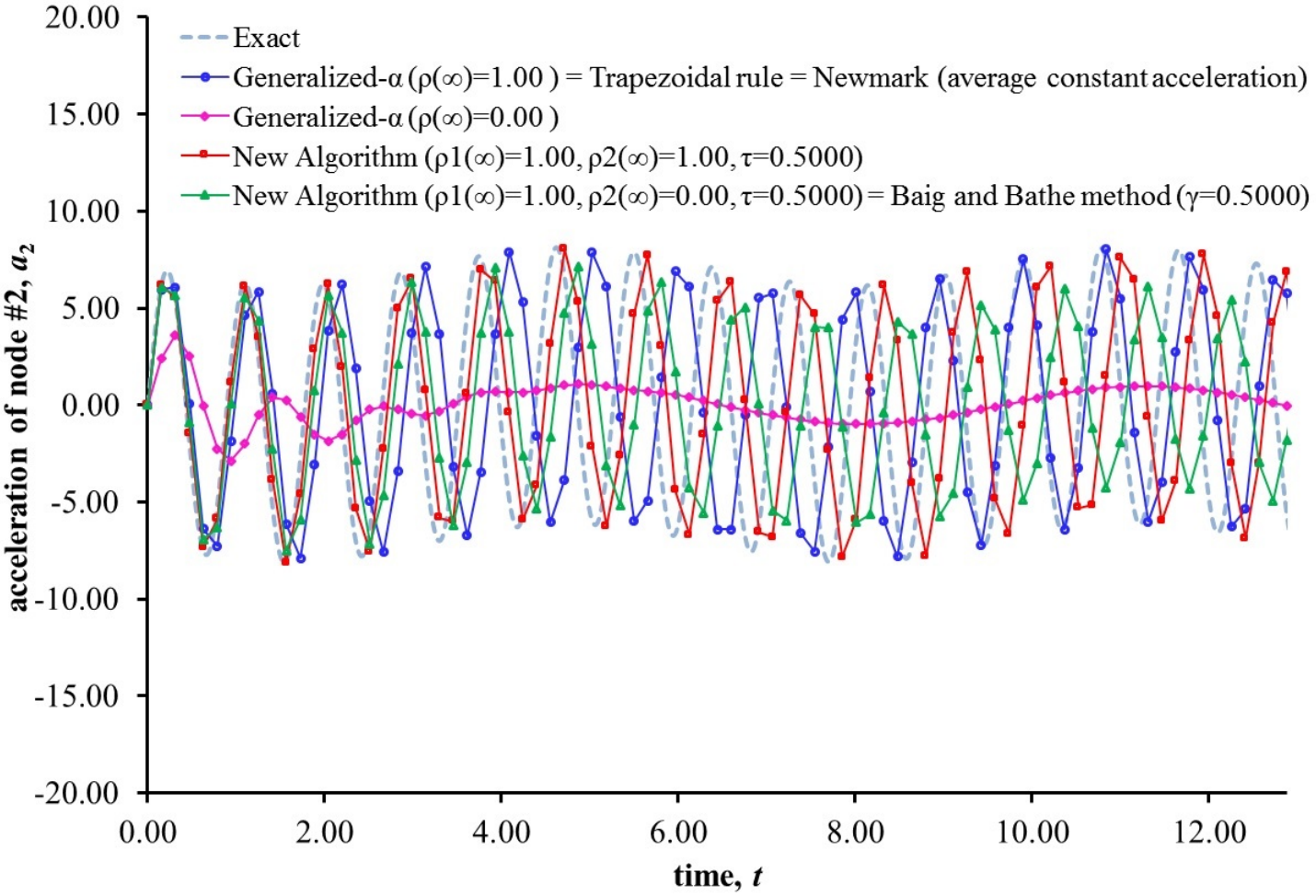

Figure 2.21: Accelerations of node 2 (a_2) with $\Delta t = T_p/40$

1. Parameters that should be selected by user:
   $0.5 \leq \tau < 1.0$ (usually $\tau = 0.5$),
   $0.0 \leq \rho_1 \leq 1.0$ (usually $\rho_1 = 1.0$),
   $0.0 \leq \rho_2 \leq 1.0$ (Here, ρ_2 can be considered as ρ_∞.)
   And note that ρ_1 and ρ_2 are used in place of $\rho_1(\infty)$ and $\rho_2(\infty)$, respectively, for simplicity.

2. Parameters that can be calculated by selecting τ, ρ_1 and ρ_2:
   $\theta_1 = \frac{1}{1+\rho_1}$,
   $\theta_2 = \frac{\tau^2\theta_1(\rho_2-1)+1}{2\{1-\tau\theta_1(1-\rho_2)\}} + \frac{\sqrt{\tau^4{\theta_1}^2(\rho_2-1)^2-4\tau^2(1-\tau){\theta_1}^2(\rho_2-1)+2\tau^2\theta_1(\rho_2+1)-4\tau\theta_1+1}}{2\{1-\tau\theta_1(1-\rho_2)\}}$

3. Algorithm coefficients:
   (The fist sub-step)
   $c_1 = \frac{1}{\tau\theta_1\Delta t}, \quad c_2 = -\frac{1}{\tau\theta_1\Delta t}, \quad c_3 = \frac{\theta_1-1}{\theta_1}$
   $c_4 = {c_1}^2, \quad c_5 = c_1, \quad c_6 = -c_1 c_2, \quad c_7 = -\left(c_1 c_3 + c_2\right)$
   $c_8 = -c_3, \quad c_9 = -c_2, \quad c_{10} = -c_3$

   (The second sub-step)
   $d_1 = \frac{\tau-2\theta_2}{\theta_2(\tau-\theta_2)\Delta t}, \quad d_2 = \frac{2\theta_2-1}{\tau\theta_2(\tau-\theta_2)\Delta t}, \quad d_3 = \frac{(1-\tau)(\tau+1-2\theta_2)}{\tau\theta_2(\tau-\theta_2)\Delta t}$
   $d_4 = \frac{\theta_2-1}{\tau(\theta_2-\tau)}, \quad d_5 = \frac{(\theta_2-1)(\tau-1)}{\tau\theta_2}$
   $d_6 = {d_1}^2, \quad d_7 = d_1, \quad d_8 = -d_1 d_3$
   $d_9 = -d_1 d_2, \quad d_{10} = -(d_1 d_5 + d_3), \quad d_{11} = -(d_1 d_4 + d_2)$
   $d_{12} = -d_5, \quad d_{13} = -d_4, \quad d_{14} = -d_3$
   $d_{15} = -d_2, \quad d_{16} = -d_5, \quad d_{17} = -d_4$

Table 2.1: Summary of parameters and coefficients.

$$\rho h \frac{\partial^2 v_0}{\partial t^2} - \frac{\partial N_{xy}}{\partial x} - \frac{\partial N_{yy}}{\partial y} = 0 \tag{2.52b}$$

$$\rho h \frac{\partial^2 w_0}{\partial t^2} - \frac{\partial Q_x}{\partial x} - \frac{\partial Q_y}{\partial y} - \left[\frac{\partial}{\partial x}\left(N_{xx}\frac{\partial w_0}{\partial x} + N_{xy}\frac{\partial w_0}{\partial y} \right) + \frac{\partial}{\partial y}\left(N_{xy}\frac{\partial w_0}{\partial x} + N_{yy}\frac{\partial w_0}{\partial y} \right) \right] - q = 0 \tag{2.52c}$$

$$\frac{\rho h^3}{12}\frac{\partial^2 \phi_x}{\partial t^2} - \frac{\partial M_{xx}}{\partial x} - \frac{\partial M_{xy}}{\partial y} + Q_x = 0 \tag{2.52d}$$

$$\frac{\rho h^3}{12}\frac{\partial^2 \phi_y}{\partial t^2} - \frac{\partial M_{xy}}{\partial x} - \frac{\partial M_{yy}}{\partial y} + Q_y = 0 \tag{2.52e}$$

where u_0, v_0 and w_0 are the mid-plane displacement pointing x, y and z directions respectively, and ϕ_x and ϕ_y are the rotations of a transverse line about the y and x axes. The plate stress resultants are defined by

$$N_{xx} = A_{11}\frac{\partial u_0}{\partial x} + A_{12}\frac{\partial v_0}{\partial y} + \frac{1}{2}\left[A_{11}\left(\frac{\partial w_0}{\partial x}\right)^2 + A_{12}\left(\frac{\partial w_0}{\partial y}\right)^2\right] \tag{2.53a}$$

$$N_{yy} = A_{12}\frac{\partial u_0}{\partial x} + A_{22}\frac{\partial v_0}{\partial y} + \frac{1}{2}\left[A_{12}\left(\frac{\partial w_0}{\partial x}\right)^2 + A_{22}\left(\frac{\partial w_0}{\partial y}\right)^2\right] \tag{2.53b}$$

$$N_{xy} = A_{66}\left(\frac{\partial u_0}{\partial y} + \frac{\partial v_0}{\partial x} + \frac{\partial w_0}{\partial x}\frac{\partial w_0}{\partial y}\right) \tag{2.53c}$$

$$M_{xy} = D_{66}\left(\frac{\partial \phi_x}{\partial y} + \frac{\partial \phi_y}{\partial x}\right) \tag{2.53d}$$

$$M_{xx} = D_{11}\frac{\partial \phi_x}{\partial x} + D_{12}\frac{\partial \phi_y}{\partial y} \tag{2.53e}$$

$$M_{yy} = D_{12}\frac{\partial \phi_x}{\partial x} + D_{22}\frac{\partial \phi_y}{\partial y} \tag{2.53f}$$

$$Q_x = A_{55}\left(\phi_x + \frac{\partial w_0}{\partial x}\right) \tag{2.53g}$$

$$Q_y = A_{44}\left(\phi_y + \frac{\partial w_0}{\partial y}\right) \tag{2.53h}$$

and the coefficients of the isotropic FSDT plate are defined by

$$\begin{aligned} &A_{11} = A_{22} = \frac{hE}{1-\nu^2}, \quad A_{12} = \frac{h\nu E}{1-\nu^2}, \quad A_{66} = hG, \quad A_{44} = A_{45} = hK_sG \\ &D_{11} = D_{22} = \frac{h^3 E}{12(1-\nu^2)}, \quad D_{12} = \frac{h^3 \nu E}{12(1-\nu^2)}, \quad D_{66} = \frac{h^3 G}{12} \end{aligned} \tag{2.54}$$

where E is Young's modulus, G is the shear modulus, ν is Poisson's ratio, $K_s = 5/6$ is the shear correction coefficient, and h is the thickness of the plate. The geometry and the boundary conditions are presented in Fig. 2.22. We take the geometric and material parameters for the problem as $a = b = 50.0\text{cm}$, $h = 0.5\,\text{cm}$, $E = 2.1 \times 10^6$ N /cm^2, $\rho = 8.0 \times 10^{-6}\ \text{N}\,\text{s}^2/\text{cm}^4$ and $\nu = 0.25$. And the initial conditions are taken as

$$\begin{aligned} &u_0(x,y,0) = v_0(x,y,0) = w_0(x,y,0) = \phi_x(x,y,0) = \phi_y(x,y,0) = 0 \\ &\dot{u}_0(x,y,0) = \dot{v}_0(x,y,0) = \dot{w}_0(x,y,0) = \dot{\phi}_x(x,y,0) = \dot{\phi}_y(x,y,0) = 0 \\ &\ddot{u}_0(x,y,0) = \ddot{v}_0(x,y,0) = \ddot{w}_0(x,y,0) = \ddot{\phi}_x(x,y,0) = \ddot{\phi}_y(x,y,0) = 0 \end{aligned} \tag{2.55}$$

We use uniformly distributed load with intensity $q = 100.0 \times H(t)\,\text{N/cm}^2$, where $H(t)$ denotes the Heaviside step function. Using the biaxial symmetry, one quadrant of the plate is chosen as the computational domain as shown in Fig. 2.22. We use 4×4 mesh of quadratic elements and specific terms of the matrices and force vectors of the finite element model are provided in Ref. [4]. We use $2\Delta t$ in the new algorithm and Δt for the trapezoidal rule for fair comparison, and $\mathbf{u}_{s+\tau}$ of the new algorithm was computed from the finite element relation given in Eq. (2.27a). The nonlinear solutions in each time step are obtained using Newton's method. Note that we used only three cases of the new algorithm (i.e., case 1: $\rho_1(\infty) = 1.0,\ \rho_2(\infty) = 0.0$ and $\tau = 0.5$, case 2: $\rho_1(\infty) = 1.0,\ \rho_2(\infty) = 0.0$ and $\tau = 0.8750$ and case 3: $\rho_1(\infty) = 1.0,\ \rho_2(\infty) = 0.9$ and $\tau = 0.5$). The nonlinear convergence criterion of $\epsilon = 10^{-4}$, as given in Ref. [4], was used. With the choice of $h = 0.5\,\text{cm}$, the plate becomes rather thin (i.e., $a/h = 100$) and in-plane terms (A_{ij}) in the stiffness matrix become larger than those of bending (D_{ij}). If h becomes smaller, making the plate thinner, the system can become even more unstable as observed in the linear three

degree-of-freedom presented in Ref. [8]. In general, some stable schemes become unstable for nonlinear systems for large time steps, and choice of smaller h can even worsen the stability. This discussion can be verified by the numerical results of the nonlinear plate bending problem shown in the figures. Three cases of the new algorithm yielded stable solutions as expected, while the solution obtained from the trapezoidal rule became unstable after roughly $10,000\,\mu$s for $\Delta t = 200\,\mu$s, as shown in Figs. 2.23, 2.25, and 2.27. The case of $\rho_2(\infty) = 0.9$ provided the displacement solution with smallest decay of amplitude for $\Delta t = 200\,\mu$s among the schemes tested, while case with $\tau = 0.8750$ showed moderate decay of amplitude and largest period elongation. It can be said that proper use of algorithmic dissipation can improved the quality of the solution when large time interval is chosen. In this case roughly one fourth of the first nonlinear period was used as the size of Δt. Thus, our analysis conducted with a single-degree-of-freedom problem still agrees with the result of the nonlinear 2-D FSDT plate bending problem. As the time step decreases, the difference between the solutions become negligible as shown in Fig. 2.24, while the amplitude of the accelerations and velocities increased as the time step decreases and they heavily depended on the scheme selected. The results are shown in Figs. 2.26, and 2.28. Note that the new algorithm used twice the size of the time step used in the trapezoidal rule (average constant acceleration method) for a fair comparison.

The specific computational procedure of the nonlinear analysis is provided in Table 2.1 and 2.3.

2.5 Conclusions

In this study, a new family of one- and two-step time integration schemes have been developed based on time collocation finite element approach, and combined using the strategy of the Baig and Bathe method [35, 40] to form a set of complete

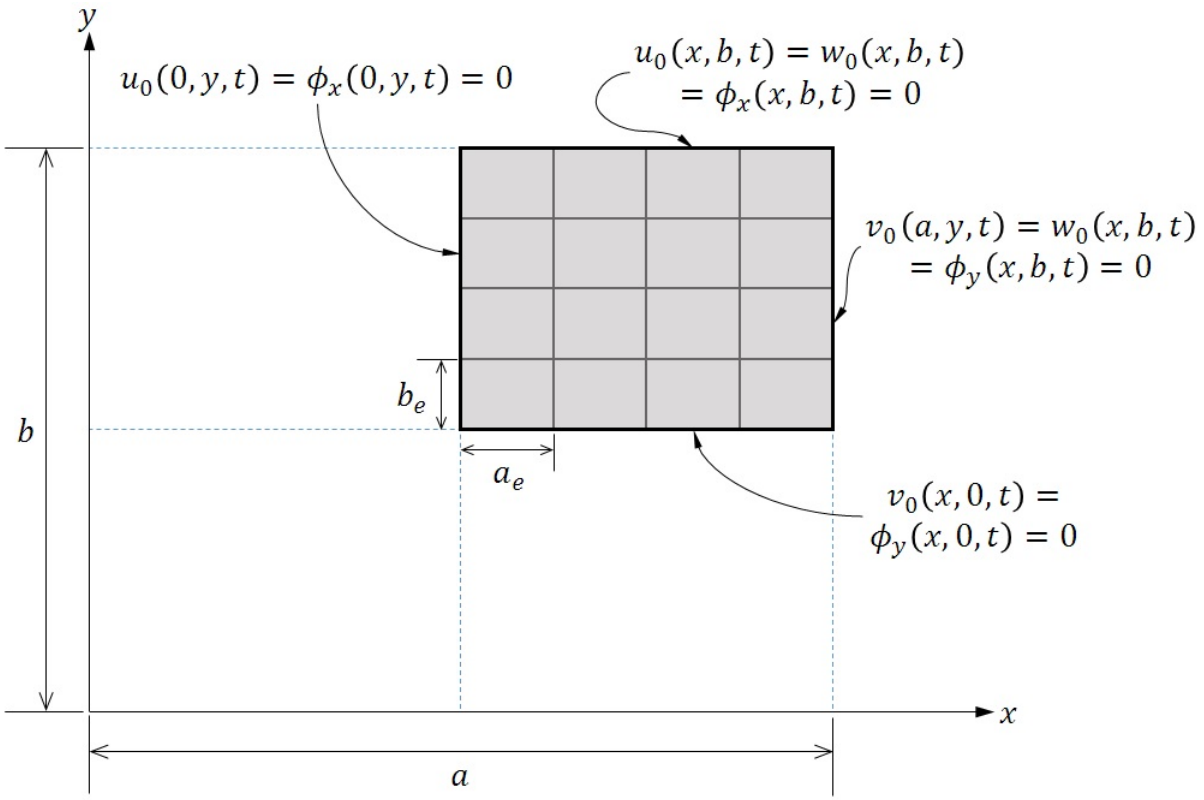

Figure 2.22: Computational domain and mesh used for nonlinear plate bending problem.

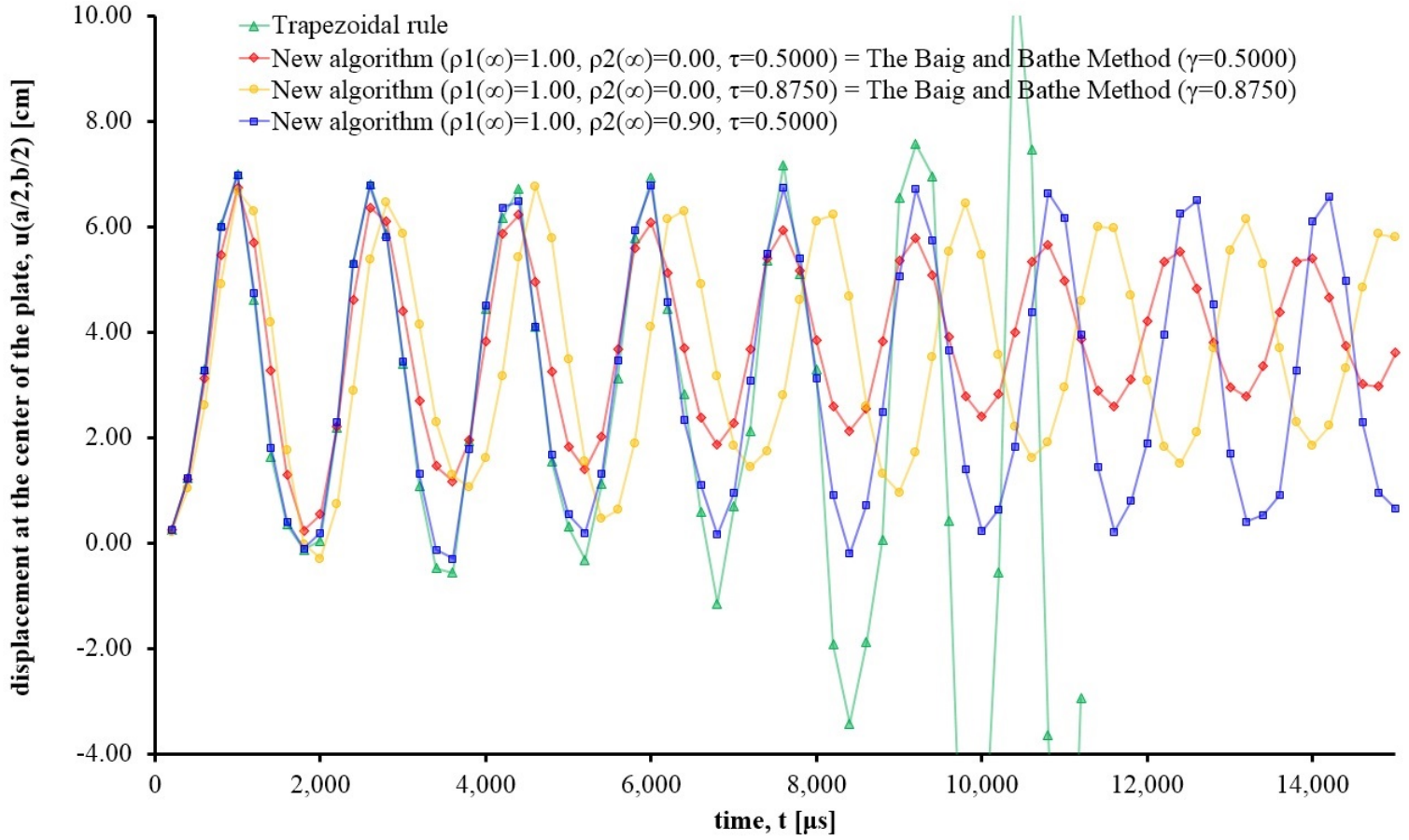

Figure 2.23: Comparison of center displacements of nonlinear FSDT plate bending problem for $\Delta T = 200\mu$s ($\Delta T = 400\mu$s for the new algorithm).

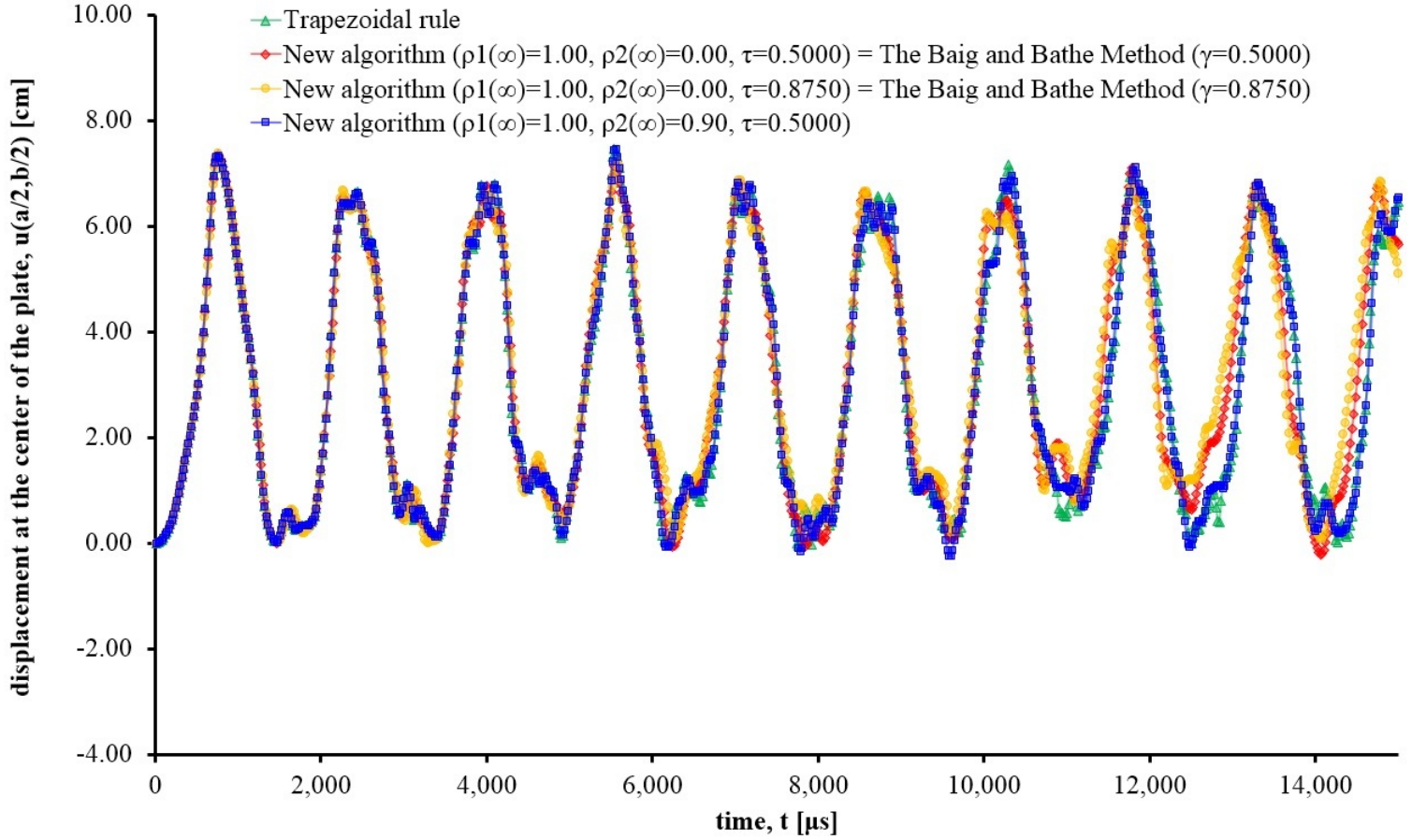

Figure 2.24: Comparison of center displacement of nonlinear FSDT plate bending problem for $\Delta T = 20\mu$s ($\Delta T = 40\mu$s for the new algorithm).

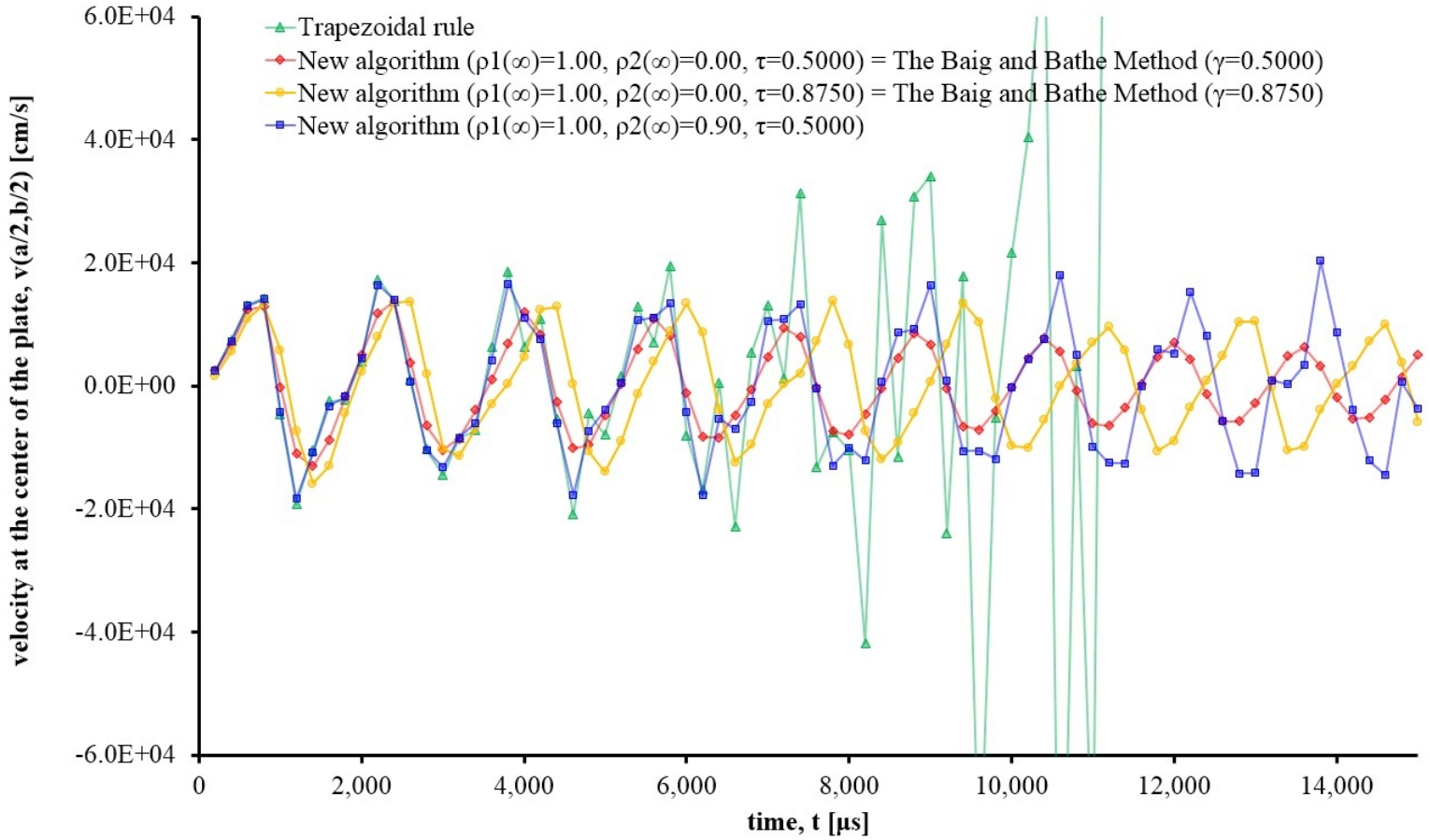

Figure 2.25: Comparison of center velocities of nonlinear FSDT plate bending problem for $\Delta T = 200\mu$s ($\Delta T = 400\mu$s for the new algorithm).

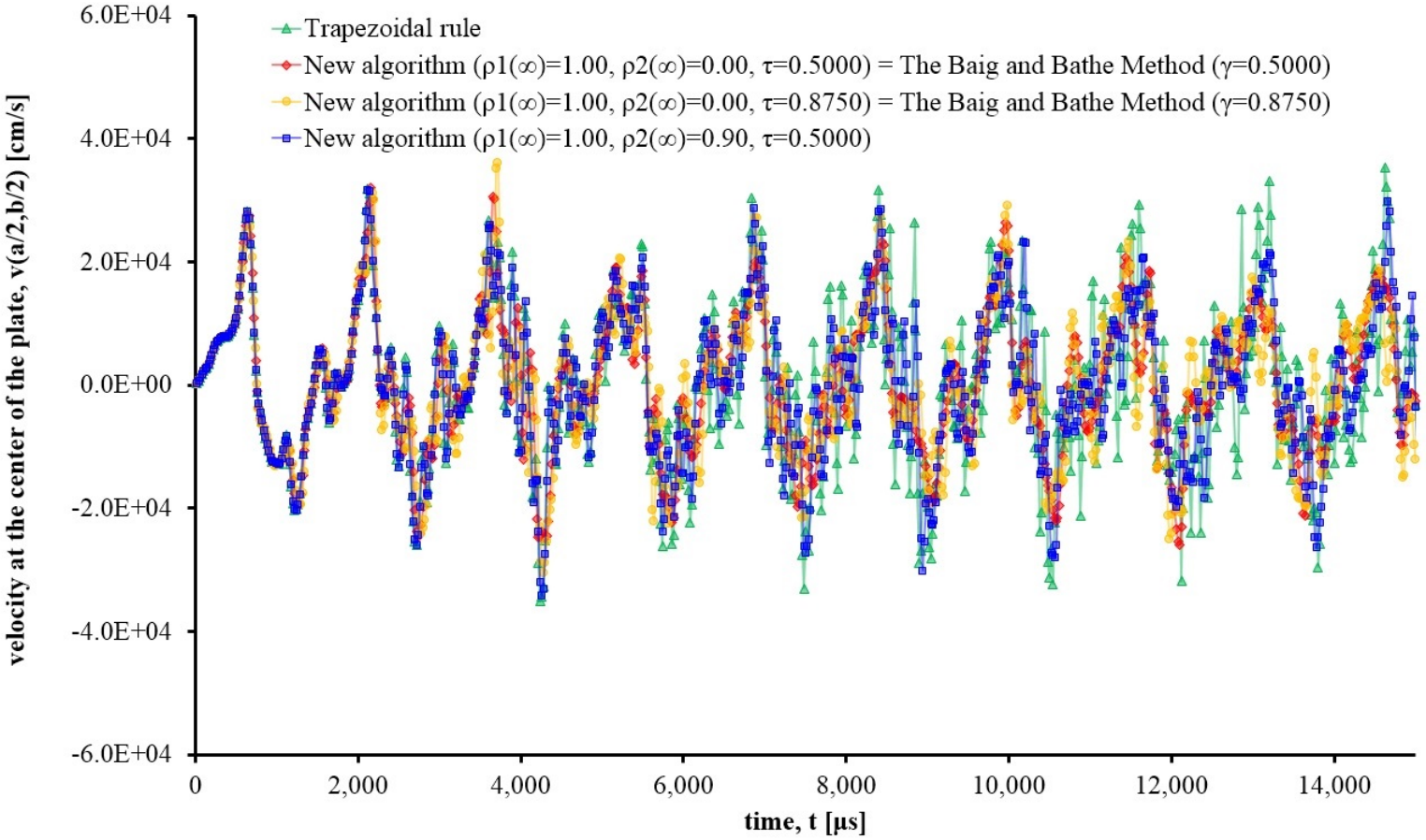

Figure 2.26: Comparison of center velocity of nonlinear FSDT plate bending problem for $\Delta T = 20\mu$s ($\Delta T = 40\mu$s for the new algorithm).

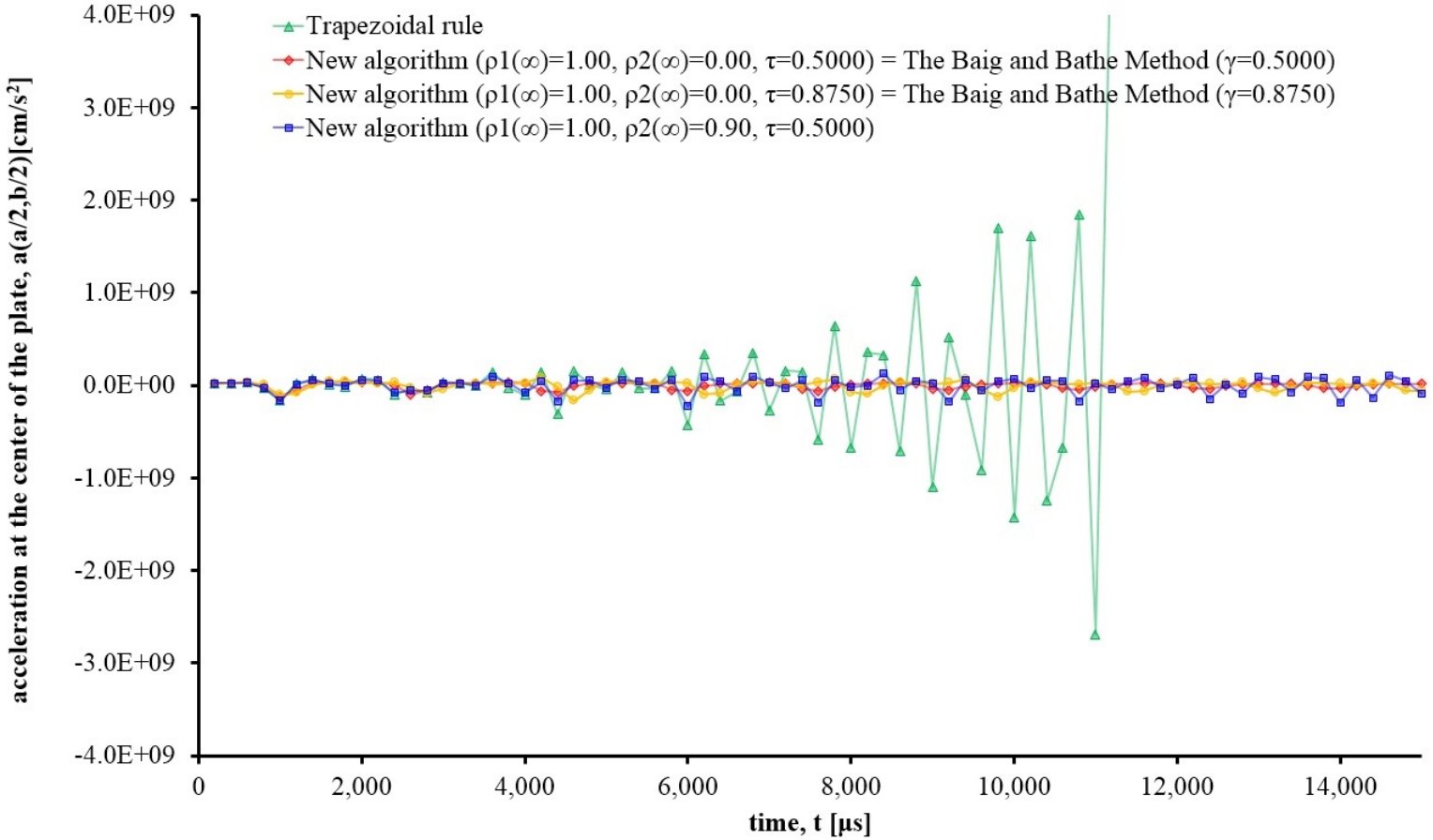

Figure 2.27: Comparison of center accelerations of nonlinear FSDT plate bending problem for $\Delta T = 200\mu$s ($\Delta T = 400\mu$s for the new algorithm).

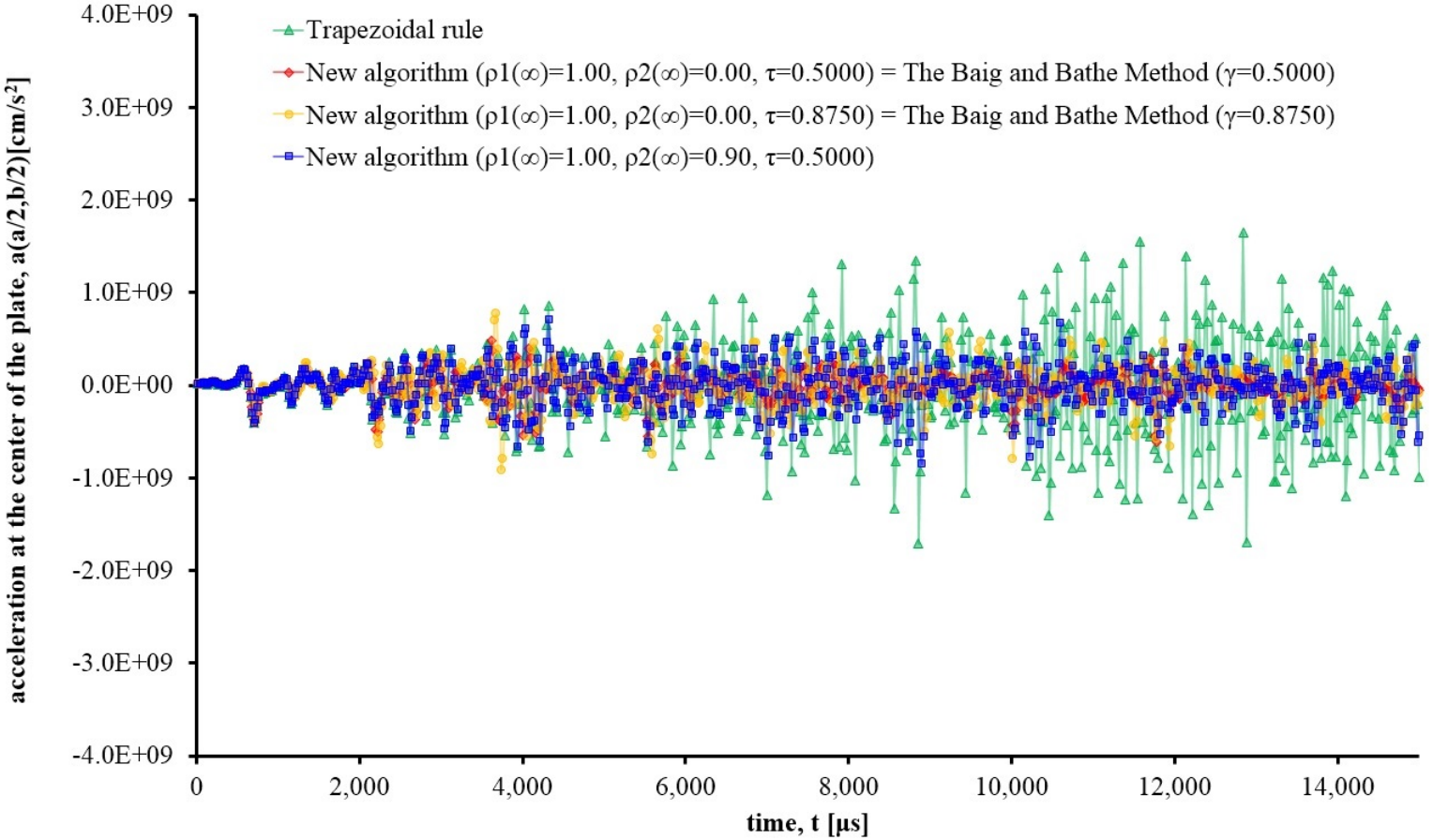

Figure 2.28: Comparison of center accelerations of nonlinear FSDT plate bending problem for $\Delta T = 20\mu$s ($\Delta T = 40\mu$s for the new algorithm).

algorithms which can be used to analyze linear and nonlinear structural dynamic problems. Three algorithmic parameters are used to synthesize desired characteristics of the algorithm. The analysis of the new algorithm has been conducted to relate its parameters to the desired dissipation level in the high frequency limit. The new algorithm is unconditionally stable and has second-order accuracy (The second order accuracy is obtained only for the choice of $\theta_1 = 0.5$). The new algorithm is able to control the dissipation in the high frequency limit while minimizing dissipation in the low frequency ranges. The new algorithm showed better numerical performance than the generalized-α method when it was analyzed and tested with linear and nonlinear problems.

A linear spring problem, specially modified from Bathe and Noh[8], was used to demonstrate the potential misuse of any asymptotic annihilation algorithm. Detecting distortions in numerical solutions was not easy when the asymptotic annihilation algorithm was used alone. In our example, the no-dissipation cases of the generalized-α method and the new algorithm detected the important high frequency while asymptotic annihilation cases of them eliminated them when relatively large time step was used. Thus use of various dissipation levels can prevent exclusion of important frequency modes, and provide better indications for the solution refinement. However, even with considerably small time step, the amplitude decay was noticeable in asymptotic annihilation cases. Some special schemes chosen from the new algorithm could stabilize the nonlinear plate bending problem and provide reasonably accurate numerical solutions with relatively large time step when the trapezoidal rule becomes unstable.

Some special features of the new algorithm are that it can include the Baig and Bathe method [35, 40] and provide a scheme that performs as no-dissipation case (almost identical to the trapezoidal rule with half Δt) as special cases. We also

emphasize that the improved performance of the algorithm is not due to increased computational cost when it is compared with the generalized-α method and the trapezoidal rule.

At the beginning of the computation

1. Evaluate $\mathbf{K}$, $\mathbf{C}$, $\mathbf{M}$ and $\mathbf{f}_0$.
2. Using $\mathbf{u}_0 = \mathbf{u}(0)$, $\mathbf{v}_0 = \mathbf{v}(0)$ and $\mathbf{a}_0 = \mathbf{a}(0)$.
3. Select parameters and compute constants. (See table 2.1 for details.)

For each increment of the time step

First sub-step

1. Impose boundary conditions, and compute $\mathbf{u}_{s+\tau}$ from
   ${}^1\hat{\mathbf{K}}\mathbf{u}_{s+\tau} = {}^1\hat{\mathbf{f}}$,
   where
   ${}^1\hat{\mathbf{K}} = c_4\mathbf{M} + c_5\mathbf{C} + \mathbf{K}$
   ${}^1\hat{\mathbf{f}} = \mathbf{M}\left(c_6\mathbf{u}_s + c_7\mathbf{v}_s + c_8\mathbf{a}_s\right) + \mathbf{C}\left(c_9\mathbf{u}_s + c_{10}\mathbf{v}_s\right) + \mathbf{f}_{s+\tau}$
2. Update $\mathbf{v}_{s+\tau}$ and $\mathbf{a}_{s+\tau}$ as
   $\mathbf{v}_{s+\tau} = c_1\mathbf{u}_{s+\tau} + c_2\mathbf{u}_s + c_3\mathbf{v}_s$
   $\mathbf{a}_{s+\tau} = c_1\mathbf{v}_{s+\tau} + c_2\mathbf{v}_s + c_3\mathbf{a}_s$

Second sub-step (Use $\mathbf{u}_{s+\tau}$, $\mathbf{v}_{s+\tau}$ and $\mathbf{a}_{s+\tau}$ obtained in the first sub-step.)

1. Impose boundary conditions, and compute $\mathbf{u}_{s+1}$ from
   ${}^2\hat{\mathbf{K}}\ \mathbf{u}_{s+1} = {}^2\hat{\mathbf{f}}$,
   where
   ${}^2\hat{\mathbf{K}} = d_6\mathbf{M} + d_7\mathbf{C} + \mathbf{K}$
   ${}^2\hat{\mathbf{f}} = \mathbf{M}\left(d_8\mathbf{u}_s + d_9\mathbf{u}_{s+\tau} + d_{10}\mathbf{v}_s + d_{11}\mathbf{v}_{s+\tau} + d_{12}\mathbf{a}_s + d_{13}\mathbf{a}_{s+\tau}\right)$
   $+\mathbf{C}\left(d_{14}\mathbf{u}_s + d_{15}\mathbf{u}_{s+\tau} + d_{16}\mathbf{v}_s + d_{17}\mathbf{v}_{s+\tau}\right) + \mathbf{f}_{s+1}$
2. Update $\mathbf{v}_{s+1}$ and $\mathbf{a}_{s+1}$ as follows:
   $\mathbf{v}_{s+1} = d_1\mathbf{u}_{s+1} + d_2\mathbf{u}_{s+\tau} + d_3\mathbf{u}_s + d_4\mathbf{v}_{s+\tau} + d_5\mathbf{v}_s$
   $\mathbf{a}_{s+1} = d_1\mathbf{v}_{s+1} + d_2\mathbf{v}_{s+\tau} + d_3\mathbf{v}_s + d_4\mathbf{a}_{s+\tau} + d_5\mathbf{a}_s$

Table 2.2: Summary of the new algorithm for linear structural system.

At the beginning of the computation

1. Evaluate $\mathbf{K}_0$, $\mathbf{C}$, $\mathbf{M}$ and $\mathbf{f}_0$.
2. Using $\mathbf{u}_0 = \mathbf{u}(0)$, $\mathbf{v}_0 = \mathbf{v}(0)$, find $\mathbf{a}_0 = \mathbf{a}(0)$.
3. Calculate parameters and constants. (See table 2.1 for details.)

For each time increment

First sub-step

1. Impose boundary conditions, and compute $\mathbf{u}_{s+\tau}^1 = {}^1\hat{\mathbf{K}}\left(\mathbf{u}_{s+\tau}^0\right)^{-1}\ {}^1\hat{\mathbf{f}}$, where $\mathbf{u}_{s+\tau}^0$ is the initial guess solution of the first sub-step and
   ${}^1\hat{\mathbf{K}}\left(\mathbf{u}_{s+\tau}\right) = c_4\mathbf{M} + c_5\mathbf{C} + \mathbf{K}\left(\mathbf{u}_{s+\tau}\right)$
   ${}^1\hat{\mathbf{f}} = \mathbf{M}\left(c_6\mathbf{u}_s + c_7\mathbf{v}_s + c_8\mathbf{a}_s\right) + \mathbf{C}\left(c_9\mathbf{u}_s + c_{10}\mathbf{v}_s\right) + \mathbf{f}_{s+\tau}$
2. For $r \geq 1$, Impose boundary conditions,
   and compute incremental solution at (r+1)th iteration
   $\delta\mathbf{u}_{s+\tau}^{r+1} = {}^1\hat{\mathbf{T}}\left(\mathbf{u}_{s+\tau}^r\right)^{-1}{}^1\hat{\mathbf{r}}\left(\mathbf{u}_{s+\tau}^r\right)$ and update total solution as
   $\mathbf{u}_{s+\tau}^{r+1} = \mathbf{u}_{s+\tau}^r + \delta\mathbf{u}_{s+\tau}^{r+1}$
   where ${}^1\hat{\mathbf{T}}\left(\mathbf{u}_{s+\tau}\right) = \frac{\partial {}^1\hat{\mathbf{r}}(\mathbf{u}_{s+\tau})}{\partial \mathbf{u}_{s+\tau}}$ and ${}^1\hat{\mathbf{r}}\left(\mathbf{u}_{s+\tau}\right) = {}^1\hat{\mathbf{f}} - {}^1\hat{\mathbf{K}}\left(\mathbf{u}_{s+\tau}\right)\mathbf{u}_{s+\tau}$.
3. Repeat 2. until converged $\mathbf{u}_{s+\tau}$ ($\approx \mathbf{u}_{s+\tau}^{r+1}$) is obtained.
4. Update $\mathbf{v}_{s+\tau}$ and $\mathbf{a}_{s+\tau}$ as
   $\mathbf{v}_{s+\tau} = c_1\mathbf{u}_{s+\tau} + c_2\mathbf{u}_s + c_3\mathbf{v}_s$
   $\mathbf{a}_{s+\tau} = c_1\mathbf{v}_{s+\tau} + c_2\mathbf{v}_s + c_3\mathbf{a}_s$

Second sub-step(Use $\mathbf{u}_{s+\tau}$, $\mathbf{v}_{s+\tau}$ and $\mathbf{a}_{s+\tau}$ obtained in the first sub-step.)

1. Impose boundary conditions, and compute $\mathbf{u}_{s+1}^1 = {}^2\hat{\mathbf{K}}\left(\mathbf{u}_{s+1}^0\right)^{-1}\ {}^2\hat{\mathbf{f}}$, where $\mathbf{u}_{s+1}^0$ is the initial guess solution of the second sub-step and
   ${}^2\hat{\mathbf{K}}\left(\mathbf{u}_{s+1}^0\right) = d_6\mathbf{M} + d_7\mathbf{C} + \mathbf{K}\left(\mathbf{u}_{s+1}\right)$
   ${}^2\hat{\mathbf{f}} = \mathbf{M}\left(d_8\mathbf{u}_s + d_9\mathbf{u}_{s+\tau} + d_{10}\mathbf{v}_s + d_{11}\mathbf{v}_{s+\tau} + d_{12}\mathbf{a}_s + d_{13}\mathbf{a}_{s+\tau}\right)$
   $+\mathbf{C}\left(d_{14}\mathbf{u}_s + d_{15}\mathbf{u}_{s+\tau} + d_{16}\mathbf{v}_s + d_{17}\mathbf{v}_{s+\tau}\right) + \mathbf{f}_{s+1}$
2. For $r \geq 1$, impose boundary conditions,
   and compute incremental solution at (r+1)th iteration
   $\delta\mathbf{u}_{s+1}^{r+1} = {}^2\hat{\mathbf{T}}\left(\mathbf{u}_{s+1}^r\right)^{-1}{}^2\hat{\mathbf{r}}\left(\mathbf{u}_{s+1}^r\right)$ and update total solution as
   $\mathbf{u}_{s+1}^{r+1} = \mathbf{u}_{s+1}^r + \delta\mathbf{u}_{s+1}^{r+1}$
   where ${}^2\hat{\mathbf{T}}\left(\mathbf{u}_{s+1}\right) = \frac{\partial^2 \hat{\mathbf{r}}(\mathbf{u}_{s+1})}{\partial \mathbf{u}_{s+1}}$ and ${}^2\hat{\mathbf{r}}\left(\mathbf{u}_{s+1}\right) = {}^2\hat{\mathbf{f}} - {}^2\hat{\mathbf{K}}\left(\mathbf{u}_{s+1}\right)\mathbf{u}_{s+1}$.
3. Repeat 2. until converged $\mathbf{u}_{s+1}$ ($\approx \mathbf{u}_{s+1}^{r+1}$) is obtained.
4. Update $\mathbf{v}_{s+1}$ and $\mathbf{a}_{s+1}$ as follows:
   $\mathbf{v}_{s+1} = d_1\mathbf{u}_{s+1} + d_2\mathbf{u}_{s+\tau} + d_3\mathbf{u}_s + d_4\mathbf{v}_{s+\tau} + d_5\mathbf{v}_s$
   $\mathbf{a}_{s+1} = d_1\mathbf{v}_{s+1} + d_2\mathbf{v}_{s+\tau} + d_3\mathbf{v}_s + d_4\mathbf{a}_{s+\tau} + d_5\mathbf{a}_s$

Table 2.3: Summary of algorithm for nonlinear structural system.

3. TIME FINITE ELEMENT METHOD II

3.1 Introduction

The equation of structural dynamics can be written as

$$\mathbf{M}\ddot{\mathbf{u}}(t) + \mathbf{C}\dot{\mathbf{u}}(t) + \mathbf{K}\mathbf{u}(t) = \mathbf{f}(t) \tag{3.1}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{C}$ is the viscous damping matrix, $\mathbf{K}$ is the stiffness matrices, $\mathbf{f}(t)$ is the vector of applied forces, $\mathbf{u}(t)$ is the displacement vector, $\dot{\mathbf{u}}(t)$ is the velocity vector, and $\ddot{\mathbf{u}}(t)$ is the acceleration vector. A solution of the initial value problem described by Eq. (3.1) satisfies the following initial conditions

$$\mathbf{u}(0) = \mathbf{u}_0 \tag{3.2a}$$

$$\dot{\mathbf{u}}(0) = \mathbf{v}_0 \tag{3.2b}$$

where $\mathbf{u}_0$ and $\mathbf{v}_0$ are the initial displacement and velocity vectors, respectively.

Recently, several higher-order time integration algorithms [36, 60, 43] have been developed based on the weighted residual method for the analysis of linear structural dynamics described by Eqs. (3.1) and (3.2). Among the existing weighted residual method based higher-order algorithms, Fung's algorithms possess many of preferable attributes [33, 74], such as improve accuracy, controllable algorithmic dissipation and unconditional stability.

In the development of Fung's algorithms, the displacement vector was approximated in time by using the $(n+1)$th-degree polynomial to satisfy the displacement and velocity initial conditions. To satisfy the initial conditions, the constant term

of the approximation was chose as the initial displacement, and the coefficient vector of the linear term of the approximation was chose as the initial velocity. For the remaining higher-degree terms of the approximation, corresponding n unknown coefficient vectors were used. Then, the traditional weighted residual method was employed to minimize the residual vector which was defined by substituting the approximated displacement vector into the equation of structural dynamics. In Fung's algorithms, the weight parameters [16, 59] were used to rewrite the integral forms of the weighted residual statements as the algebraic forms. In the algebraic forms of the weighted residual statements, the weight parameters were optimized by using the single-degree-of-freedom problems to achieve preferable attributes. As a result of the optimization of the weight parameters, Fung's algorithms can control the full range of algorithmic dissipations in the high frequency limit through a free parameter (i.e., the spectral radius in the high frequency limit). If the displacement vector is approximated as $(n+1)$th-degree polynomial, $(2n-1)$th- and $(2n)$th-order accurate algorithms are obtained, n being the number of the unknown coefficient vectors to be determined.

As discussed in Ref. [60], however, Fung's algorithms require additional weight parameters to retain the improved order of accuracy for the particular solutions in the presence of higher-order externally applied forces. These additional weight parameters can be considered as a minor drawback of Fung's algorithms.

Idesman [43] also employed the weighted residual method to develop another family of higher-order algorithms. In the development of Idesman's algorithms, the equation of structural dynamics was rewritten as a set of two first-order equations by including the velocity vector as an additional time dependent variable. In the rewritten first-order equations (also called mixed formulations), the displacement and velocity vectors were approximated by using the equal $(n+1)$th-degree polynomial, and

the approximated displacement and velocity vectors satisfies the displacement and velocity initial conditions, respectively. Then, the two rewritten first-order equations were used to define two residual vectors in time. In the weighted residual statements, two specially constructed weighted functions were used. In Idesman's original work [43], he stated that his algorithms could provide $(2n)$th-order accuracy, which is one order higher than other equivalent (in terms of computational cost) algorithms. As a matter of fact, however, Idesman's algorithms can provide only nth-order accuracy if the parameter included in the weight functions is restated as the form suggested in Ref. [43]. This is a serious drawback of Idesman's algorithms, because other well known algorithms can provide $(2n-1)$th- or $(2n)$th-order accuracy for the same level of computational effort.

Other than the poor accuracy, Idesman's algorithms cannot control the algorithmic dissipations in the high frequency limit in a real sense. However, Idesman's algorithms can change overall profiles of spectral radii, while keeping the ultimate spectral radii [44, 33] as the same. As a result of the limited dissipation control capability, his algorithms cannot include the non-dissipative and asymptotic annihilating cases as special cases of the algorithms, which is another serious drawback in a practical view point. As he mentioned in Ref. [43], it should be noted that the algorithm can become slightly unstable near the low frequency limit, depending on the choice of the parameter included in the weight functions.

In both methods, the unknown coefficient vectors of the approximations do not have physical meanings. Due to this, the solutions at the end of the time interval should be computed by using the approximations, once these coefficient vectors are determined by solving the final form of fully discrete equations.

The purpose of this study is to eliminate some shortcomings and limitations of the existing weighted residual method based higher-order algorithms. To this end,

we approximate the displacement vector of the linear structural dynamics by using the Hermite interpolation functions in time.

With the Hermite approximation, Eq. (3.1) can be directly manipulated to define the residual vector. To reduce the number of the weight parameters in the algebraic forms of the weighted residual statements, we use two different order time derivatives of the residual vector in the modified weighted residual statements. Eq. (3.1) and its time derivatives at the time nodes are also used to eliminate the second- and higher-order nodal time derivatives included in the approximation of the displacement. Through this unique setting of computational framework, we expect our new algorithms to be more efficient and intuitive than the existing algorithms.

3.2 Development

3.2.1 Hermite Approximations in Time

In this study, general pth-degree Hermite interpolation functions [85, 92] without internal nodes are considered for the development of the new family of time integration algorithms, p being an odd integer greater than or equal to 3. The schematic concept of the time element obtained by using the pth-degree Hermite interpolation functions over the time interval ($t_s \leq t \leq t_{s+1}$) is presented in Fig. 3.1.

By using the pth-degree Hermite interpolation functions, the displacement vector $\mathbf{u}(t)$ in Eq. (3.1) can be approximation as

$$\mathbf{u}(t) \cong \tilde{\mathbf{u}}(t) = \sum_{k=0}^{\frac{p-1}{2}} \left({}_s\phi_k(t) \overset{(k)}{\mathbf{u}}_s + {}_{s+1}\phi_k(t) \overset{(k)}{\mathbf{u}}_{s+1} \right) \tag{3.3}$$

where $\overset{(k)}{\mathbf{u}}_s$ and $\overset{(k)}{\mathbf{u}}_{s+1}$ are the kth-order nodal time derivatives of the displacement vector at t_s and t_{s+1}, respectively, ${}_s\phi_k(t)$ is the Hermite interpolation function associated with $\overset{(k)}{\mathbf{u}}_s$, and ${}_{s+1}\phi_k(t)$ is the Hermite interpolation function associated with $\overset{(k)}{\mathbf{u}}_{s+1}$.

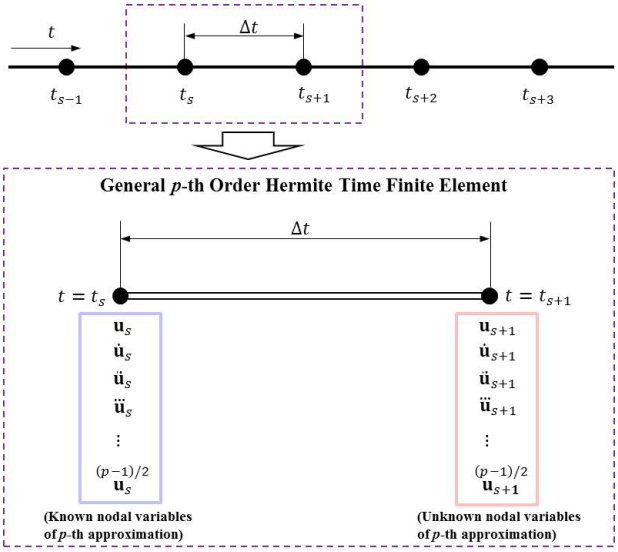

Figure 3.1: Schematic presentation of time element obtained from pth-order Hermite interpolation functions.

Here, $\overset{(k)}{\mathbf{u}}_s$ is the known property at $t = t_s$, while $\overset{(k)}{\mathbf{u}}_{s+1}$ is the unknown property at $t = t_{s+1}$. For the completeness of the development procedure, we present the Hermite interpolation functions for $p = 3, 5, 7$. For $p = 3$, the cubic Hermite interpolation functions are given by

$$
\begin{aligned}
{}_s\phi_0(t) &= 2\,\bar{t}^3 - 3\,\bar{t}^2 + 1 \\
{}_s\phi_1(t) &= (\bar{t}^3 - 2\,\bar{t}^2 + \bar{t})\Delta t \\
{}_{s+1}\phi_0(t) &= -2\,\bar{t}^3 + 3\,\bar{t}^2 \\
{}_{s+1}\phi_1(t) &= (\bar{t}^3 - \bar{t}^2)\Delta t
\end{aligned}
\tag{3.4}
$$

For $p = 5$, the quintic Hermite interpolation functions are given by

$$
\begin{aligned}
{}_s\phi_0(t) &= -6\,\bar{t}^5 + 15\,\bar{t}^4 - 10\,\bar{t}^3 + 1 \\
{}_s\phi_1(t) &= (-3\,\bar{t}^5 + 8\,\bar{t}^4 - 6\,\bar{t}^3 + \bar{t})\Delta t \\
{}_s\phi_2(t) &= \frac{1}{2}(-\,\bar{t}^5 + 3\,\bar{t}^4 - 3\,\bar{t}^3 + \,\bar{t}^2)\Delta t^2 \\
{}_{s+1}\phi_0(t) &= 6\,\bar{t}^5 - 15\,\bar{t}^4 + 10\,\bar{t}^3 \\
{}_{s+1}\phi_1(t) &= (-3\,\bar{t}^5 + 7\,\bar{t}^4 - 4\,\bar{t}^3)\Delta t \\
{}_{s+1}\phi_2(t) &= \frac{1}{2}(\bar{t}^5 - 2\,\bar{t}^4 + \bar{t}^3)\Delta t^2
\end{aligned}
\tag{3.5}
$$

For $p = 7$, the septic Hermite interpolation functions are given by

$$
\begin{aligned}
{}_s\phi_0(t) &= 20\,\bar{t}^7 - 70\,\bar{t}^6 + 84\,\bar{t}^5 - 35\,\bar{t}^4 + 1 \\
{}_s\phi_1(t) &= (10\,\bar{t}^7 - 36\,\bar{t}^6 + 45\,\bar{t}^5 - 20\,\bar{t}^4 + \bar{t})\Delta t \\
{}_s\phi_2(t) &= \frac{1}{2}(4\,\bar{t}^7 - 15\,\bar{t}^6 + 20\,\bar{t}^5 - 10\,\bar{t}^4 + \bar{t}^2)\Delta t^2 \\
{}_s\phi_3(t) &= \frac{1}{6}(\bar{t}^7 - 4\,\bar{t}^6 + \bar{t}^5 - 4\,\bar{t}^4 + \bar{t}^3)\Delta t^3 \\
{}_{s+1}\phi_0(t) &= -20\,\bar{t}^7 + 70\,\bar{t}^6 - 84\,\bar{t}^5 + 35\,\bar{t}^4 \\
{}_{s+1}\phi_1(t) &= (10\,\bar{t}^7 - 34\,\bar{t}^6 + 39\,\bar{t}^5 - 15\,\bar{t}^4)\Delta t \\
{}_{s+1}\phi_2(t) &= \frac{1}{2}(-4\,\bar{t}^7 + 13\,\bar{t}^6 - 14\,\bar{t}^5 + 5\,\bar{t}^4)\Delta t^2 \\
{}_{s+1}\phi_3(t) &= \frac{1}{6}(\bar{t}^7 - 3\,\bar{t}^6 + 3\,\bar{t}^5 - \bar{t}^4)\Delta t^3
\end{aligned}
\tag{3.6}
$$

where, $\bar{t}$ is the non-dimensional time which is defined as $\bar{t} = \frac{t - t_s}{\Delta t}$, Δt being the size of the time interval. Plots of the cubic, quintic, and septic Hermite interpolation functions are presented in Figures 3.2-3.4.

If $\tilde{\mathbf{u}}$ given in Eq. (3.3) is substituted into Eq. (3.1), the residual vector is defined

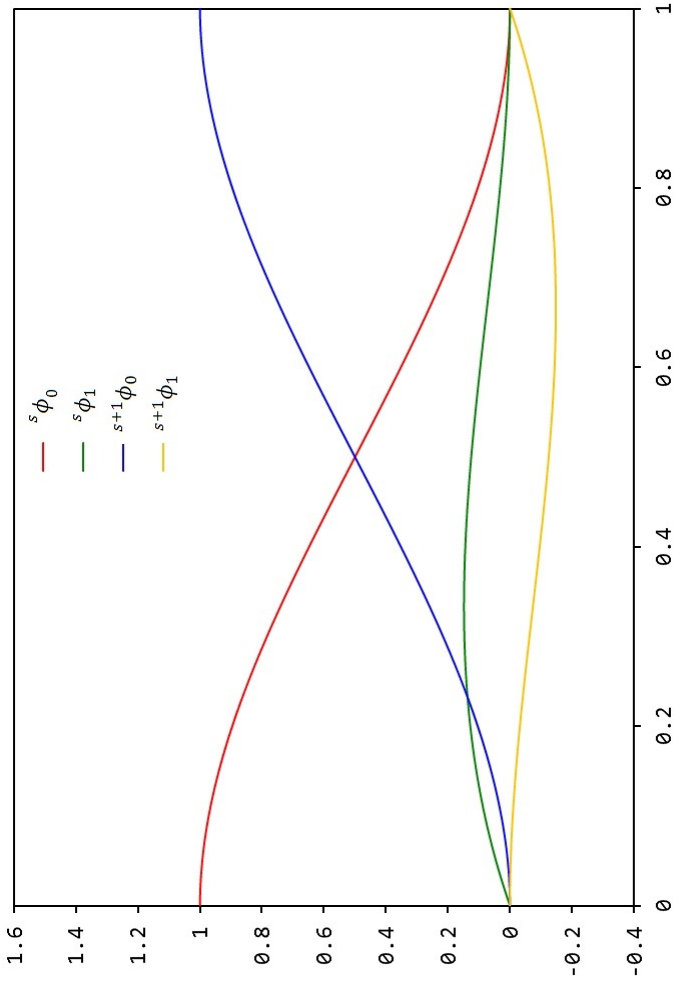

Figure 3.2: Cubic(3rd-degree) Hermite interpolation functions

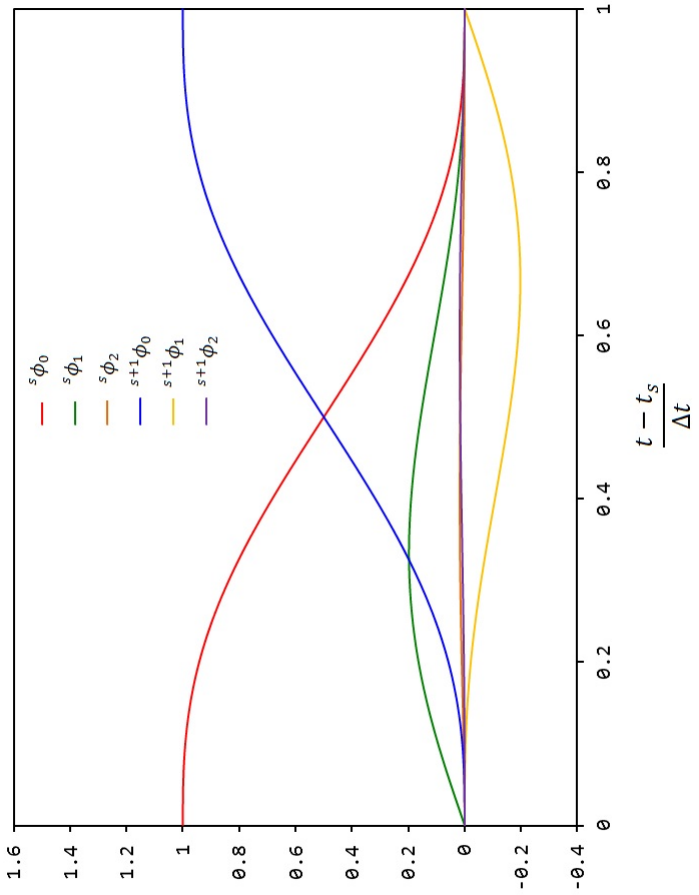

Figure 3.3: Quintic(5th-degree) Hermite interpolation functions

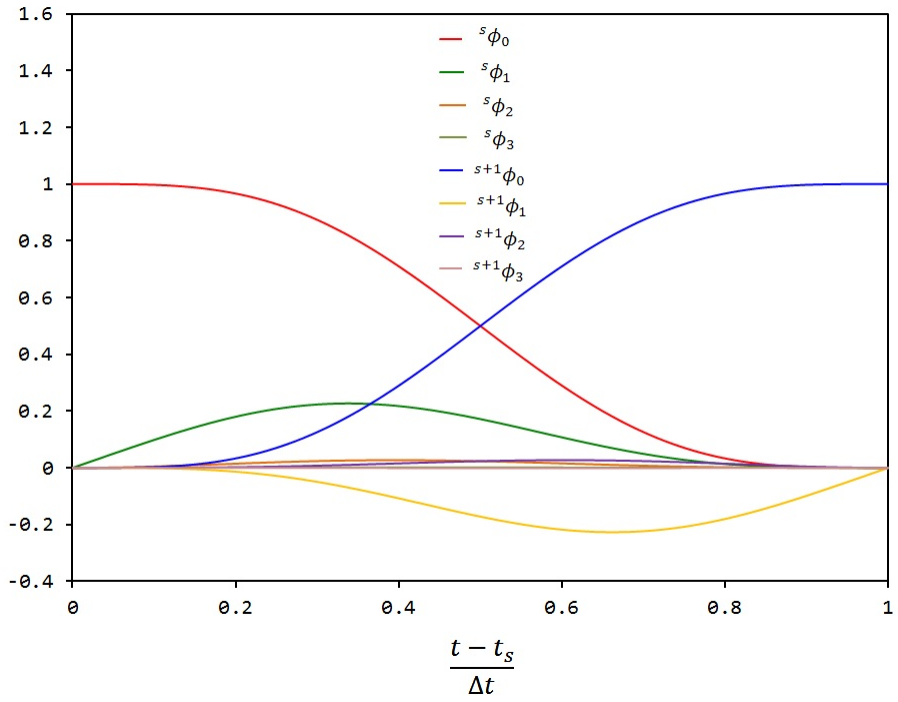

Figure 3.4: Septic(7th-degree) Hermite interpolation functions

as

$$\mathbf{r}(t) = \mathbf{M}\ddot{\tilde{\mathbf{u}}}(t) + \mathbf{C}\dot{\tilde{\mathbf{u}}}(t) + \mathbf{K}\tilde{\mathbf{u}}(t) - \tilde{\mathbf{f}}(t) \neq \mathbf{0} \tag{3.7}$$

Note that we also interpolate $\mathbf{f}(t)$ as $\tilde{\mathbf{f}}(t)$ by using the same pth-degree Hermite interpolation functions used for the approximation of $\mathbf{u}(t)$.

3.2.2 Modified Weighted Residual Statement and Dynamic Equilibrium Equations

Majority of the existing higher-order algorithms were developed based on the weighted residual method. In these algorithms, the traditional weighted residual statement is used for the minimization of the residual vector. If the traditional weighted residual statement is employed for the minimization of $\mathbf{r}(t)$ given in Eq. (3.7), $\mathbf{r}(t)$ is minimized according to

$$\mathbf{0} = \int_{t_s}^{t_{s+1}} w_i(t)\ \mathbf{r}(t)\ \mathrm{d}t \quad \text{for} \quad i = 1, 2, ..., \frac{p+1}{2} \tag{3.8}$$

where p is the degree of the Hermite interpolation functions used to approximate $\mathbf{u}(t)$, and $w_i(t)$ is the ith weight function. Naturally $\frac{p+1}{2}$ weight functions should be used to find $\frac{p+1}{2}$ unknown nodal variables included in the approximation of $\mathbf{u}(t)$. In traditional methods, total $\frac{p+1}{2} \times p$ weight parameters are needed to restate Eq. (3.8) in the algebraic form.

On the other hand, we propose two modified weight residual statements which can be stated as

$$\mathbf{0} = \int_{t_s}^{t_{s+1}} w(t)\ {}^{(p-3)}\mathbf{r}(t)\ \mathrm{d}t \tag{3.9a}$$

$$\mathbf{0} = \int_{t_s}^{t_{s+1}} w(t)\ {}^{(p-2)}\mathbf{r}(t)\ \mathrm{d}t \tag{3.9b}$$

where $w(t)$ is the weight function, and ${}^{(p-3)}\mathbf{r}(t)$ and ${}^{(p-2)}\mathbf{r}(t)$ are

$${}^{(p-3)}\mathbf{r}(t) = \frac{\mathrm{d}^{p-3}}{\mathrm{d}t^{p-3}}\mathbf{r}(t) \tag{3.10a}$$

$${}^{(p-2)}\mathbf{r}(t) = \frac{\mathrm{d}^{p-2}}{\mathrm{d}t^{p-2}}\mathbf{r}(t) \tag{3.10b}$$

Here, two linearly independent relations are obtained from the two different order differentiations of the residual vector with respect to time as given in Eqs. (3.9a) and (3.9b). In fact, Gellert [93] already manipulated the first-order time derivative of the residual vector in the minimization procedure to obtain a linearly independent relation. In Gellert's work, the cubic Hermite interpolation functions were used for the approximation of the displacement vector, and the equation of structural dynamics was used to define the residual vector. Then, the collocation method was applied to the residual vector, its first derivative, and its first and second integrals as the minimization method, and the end of interval was used as the collocation point. In some sense, our approach and Gellert's approach share the key idea of obtaining linearly independent relations, but the specific procedures are quite different.

In our case, it can be shown that the highest degree terms of t in Eqs. (3.10a) and (3.10b) are always the cubic and quadratic, respectively, regardless of the degree of the Hermite interpolation functions used. Due to this unconventional approach, we can always use only three weight parameters to restate Eqs. (3.9a) and (3.9b) in the algebraic forms, which will make the optimizations of weight parameters very simple.

There are total $\frac{p+1}{2}$ unknown nodal variables to be determined if the pth-degree Hermite interpolation functions are used. Since only two linearly dependent relations

can be obtained from Eqs. (3.9a) and (3.9b), additional $\frac{p-3}{2}$ linearly independent relations are still required. If $p = 3$, Eqs. (3.9a) and (3.9b) can provide sufficient number of linearly independent equations because only $\mathbf{u}_{s+1}$ and $\dot{\mathbf{u}}_{s+1}$ are unknown vectors included in the Hermite approximation. However, if $p \geq 5$, additional $\frac{p-3}{2}$ linearly independent relations can be obtained from

$$\frac{\mathrm{d}^k}{\mathrm{d}t^k}\left(\mathbf{M}\ddot{\mathbf{u}}(t) + \mathbf{C}\dot{\mathbf{u}}(t) + \mathbf{K}\mathbf{u}(t) - \mathbf{f}(t)\right)\Big|_{t=t_{s+1}} = 0 \quad \text{for} \quad k = 0, 1, ..., \frac{p-5}{2} \tag{3.11}$$

By using Eq. (3.11) consecutively, the second- and higher-order unknown nodal time derivatives at $t = t_{s+1}$ (i.e., $\ddot{\mathbf{u}}_{s+1}, \dddot{\mathbf{u}}_{s+1}, ..., {}^{(p-1)/2}\mathbf{u}_{s+1}$) of Eqs. (3.10a) and (3.10b) can be stated in terms of $\mathbf{u}_{s+1}$ and $\dot{\mathbf{u}}_{s+1}$. Then, $\mathbf{u}_{s+1}$ and $\dot{\mathbf{u}}_{s+1}$ can be found by solving Eqs. (3.9a) and (3.9b). It should be noted that Eq. (3.1) is exactly satisfied at $t = t_{s+1}$ through the case $k = 0$ of Eq. (3.11). Similarly, second- and higher-order known nodal time derivatives at $t = t_s$ can also be stated in terms of $\mathbf{u}_s$ and $\dot{\mathbf{u}}_s$ by using

$$\frac{\mathrm{d}^k}{\mathrm{d}t^k}\left(\mathbf{M}\ddot{\mathbf{u}}(t) + \mathbf{C}\dot{\mathbf{u}}(t) + \mathbf{K}\mathbf{u}(t) - \mathbf{f}(t)\right)\Big|_{t=t_s} = 0 \quad \text{for} \quad k = 0, 1, ..., \frac{p-5}{2} \tag{3.12}$$

3.2.3 Weight Parameters

To restate the integral forms of the weighted residual statements given in Eqs. (3.9a) and (3.9b) as the algebraic forms, only three weight parameters are required in our procedure. These three weight parameters are defined as

$$\theta_1 = \frac{\int_{t_s}^{t_{s+1}} w(t)\, t\, \mathrm{d}t}{\Delta t \int_{t_s}^{t_{s+1}} w(t)\, \mathrm{d}t} \tag{3.13a}$$

$$\theta_2 = \frac{\int_{t_s}^{t_{s+1}} w(t)\, t^2 \,\mathrm{d}t}{\Delta t^2 \int_{t_s}^{t_{s+1}} w(t)\,\mathrm{d}t} \tag{3.13b}$$

$$\theta_3 = \frac{\int_{t_s}^{t_{s+1}} w(t)\, t^3 \,\mathrm{d}t}{\Delta t^3 \int_{t_s}^{t_{s+1}} w(t)\,\mathrm{d}t} \tag{3.13c}$$

where $\Delta t = t_{s+1} - t_s$ is the size of the time interval. It should be noted that $k = 0$ will always give $\theta_0 = 1$, thus θ_0 can be regarded as 1. After eliminating the second- and higher-order nodal time derivatives in Eqs. (3.10a) and (3.10b) by using Eqs. (3.11) and (3.12) consecutively, we can restate Eqs. (3.10a) and (3.10b) as

$$\mathbf{0} = \mathbf{\Theta}_1\Big(\mathbf{M}, \mathbf{C}, \mathbf{K}, \mathbf{u}_s, \dot{\mathbf{u}}_s, \mathbf{u}_{s+1}, \dot{\mathbf{u}}_{s+1}, \Delta t, \theta_1, \theta_2, \theta_3\Big) \tag{3.14a}$$

$$\mathbf{0} = \mathbf{\Theta}_2\Big(\mathbf{M}, \mathbf{C}, \mathbf{K}, \mathbf{u}_s, \dot{\mathbf{u}}_s, \mathbf{u}_{s+1}, \dot{\mathbf{u}}_{s+1}, \Delta t, \theta_1, \theta_2\Big) \tag{3.14b}$$

where $\mathbf{\Theta}_1$ and $\mathbf{\Theta}_2$ are the coupled algebraic vectors of $m \times 1$, m being the size of Eq. (3.1).

3.2.4 Optimization

In many literatures [8, 20, 49], stability and accuracy analyses of time integration algorithms have been conducted by using the single-degree-of-freedom problem. We can also optimize three weight parameters included in Eqs. (3.14a) and (3.14b) by using a similar procedure and the single-degree-of-freedom problem. We note that the optimization of the weight parameters in the new algorithms is similar to the optimization of the weight parameters used in Ref. [59]. The single-degree-of-freedom problem is given by

$$\ddot{u}(t) + 2\,\xi\,\omega\,\dot{u}(t) + \omega^2\,u(t) = f(t) \tag{3.15}$$

with the initial conditions of

$$u(0) = u_0 \tag{3.16a}$$

$$\dot{u}(0) = v_0 \tag{3.16b}$$

where, u_0 and v_0 are the initial displacement and velocity. For the homogeneous case (i.e., $f(t) = 0$), the exact solution of Eqs. (3.15)-(3.16) is given by

$$\begin{aligned} u\left(t\right) = & \exp^{(-\xi\omega t)}\left(\cos\left(\omega_d t\right) + \frac{\xi\omega}{\omega_d}\sin\left(\omega_d t\right)\right)u_0 \\ & + \frac{1}{\omega_d}\exp^{(-\xi\omega t)}\sin\left(\omega_d t\right)v_0 \end{aligned} \tag{3.17}$$

and by differentiating $u\left(t\right)$ with respect to t once, we obtain the exact velocity solution

$$\begin{aligned} \dot{u}\left(t\right) = & -\frac{\left(\xi^2\omega^2 + {\omega_d}^2\right)}{\omega_d}\exp^{(-\xi\omega t)}\sin\left(\omega_d t\right)u_0 \\ & + \frac{1}{\omega_d}\exp^{(-\xi\omega t)}\Big(-\xi\omega\sin\left(\omega_d t\right) + \omega_d\cos\left(\omega_d t\right)\Big)v_0 \end{aligned} \tag{3.18}$$

where $\omega_d = \sqrt{1-\xi^2}\ \omega$. Now, the exact solutions given in Eqs. (3.17) and (3.18) can be used to write the exact discrete solutions at $t = \Delta t$. With the given initial conditions, the exact solution at $t = \Delta t$ can be written as

$${}^{e}\mathbf{x}_1 = {}^{e}\mathbf{A}\mathbf{x}_0 \tag{3.19}$$

where ${}^{e}\mathbf{A}$, ${}^{e}\mathbf{x}_1$ and $\mathbf{x}_0$ are the exact amplification matrix, the exact solution vector, and the initial condition vector. Here ${}^{e}\mathbf{A}$, ${}^{e}\mathbf{x}_1$ and $\mathbf{x}_0$ can be defined as

$$ {}^{e}\mathbf{x}_1 = \begin{Bmatrix} {}^{e}u(\Delta t) \\ {}^{e}\dot{u}(\Delta t) \end{Bmatrix}, \quad {}^{e}\mathbf{A} = \begin{bmatrix} {}^{e}A_{11}(\xi,\omega,\Delta t) & {}^{e}A_{12}(\xi,\omega,\Delta t) \\ {}^{e}A_{21}(\xi,\omega,\Delta t) & {}^{e}A_{22}(\xi,\omega,\Delta t) \end{bmatrix}, \quad \mathbf{x}_0 = \begin{Bmatrix} u_0 \\ v_0 \end{Bmatrix} \tag{3.20} $$

and the entries of ${}^{e}\mathbf{A}$ are given by

$$ \begin{aligned} {}^{e}A_{11} =& \exp^{(-\xi\omega\Delta t)}\left(\cos(\omega_d \Delta t) + \frac{\xi\omega}{\omega_d}\sin(\omega_d \Delta t)\right) \\ {}^{e}A_{12} =& \frac{1}{\omega_d}\exp^{(-\xi\omega\Delta t)}\sin(\omega_d \Delta t) \\ {}^{e}A_{21} =& -\frac{(\xi^2\omega^2 + {\omega_d}^2)}{\omega_d}\exp^{(-\xi\omega\Delta t)}\sin(\omega_d \Delta t) \\ {}^{e}A_{22} =& \frac{1}{\omega_d}\exp^{(-\xi\omega\Delta t)}\Big(-\xi\omega\sin(\omega_d \Delta t) + \omega_d\cos(\omega_d \Delta t)\Big) \end{aligned} \tag{3.21} $$

Here, ${}^{e}u(\Delta t)$ and ${}^{e}\dot{u}(\Delta t)$ are the exact displacement and velocity solutions at $t = \Delta t$. Similarly, the numerical solutions obtained by using the algorithms can also be written in the form of

$$ {}^{a}\mathbf{x}_1 = {}^{a}\mathbf{A}\mathbf{x}_0 \tag{3.22} $$

where ${}^{a}\mathbf{A}$ and ${}^{a}\mathbf{x}_1$ are the numerical amplification matrix and the numerical solution vector. ${}^{a}\mathbf{x}_1$, ${}^{a}\mathbf{x}_0$, and ${}^{a}\mathbf{A}$ can be defined as

$$ \begin{aligned} {}^{a}\mathbf{x}_1 &= \begin{Bmatrix} {}^{a}u(\Delta t) \\ {}^{a}\dot{u}(\Delta t) \end{Bmatrix}, \quad \mathbf{x}_0 = \begin{Bmatrix} u_0 \\ v_0 \end{Bmatrix} \\ {}^{a}\mathbf{A} &= \begin{bmatrix} {}^{a}A_{11}(\xi,\omega,\Delta t,\theta_1,\theta_2,\theta_3) & {}^{a}A_{12}(\xi,\omega,\Delta t,\theta_1,\theta_2,\theta_3) \\ {}^{a}A_{21}(\xi,\omega,\Delta t,\theta_1,\theta_2) & {}^{a}A_{22}(\xi,\omega,\Delta t,\theta_1,\theta_2) \end{bmatrix} \end{aligned} \tag{3.23} $$

Here, ${}^{a}u(\Delta t)$ and ${}^{a}\dot{u}(\Delta t)$ are the numerical displacement and velocity solutions at $t = \Delta t$. Eqs. (3.14a) and (3.14b) can be directly used to find ${}^{a}u(\Delta t)$ and ${}^{a}\dot{u}(\Delta t)$ by simply setting $\mathbf{M} = 1$, $\mathbf{C} = 2\,\xi\omega$, $\mathbf{K} = \omega^2$, $\mathbf{u}_s = u_0$, $\mathbf{v}_s = v_0$, $\mathbf{u}_{s+1} = {}^{a}u(\Delta t)$ and $\dot{\mathbf{u}}_{s+1} = {}^{a}v(\Delta t)$.

Since we arranged the exact and numerical discrete solutions in the similar forms given in Eqs. (3.19) and (3.22), the optimization of θ_1, θ_2 and θ_3 can be done by comparing the entries of ${}^{e}\mathbf{A}$ and ${}^{a}\mathbf{A}$. For a time integration algorithm to be pth-order accurate, the highest degree term of Δt in the Taylor expansion of ${}^{e}A_{ij} - {}^{a}A_{ij}$ (for $i, j = 1, 2$) should satisfy

$$ {}^{e}A_{ij} - {}^{a}A_{ij} = \mathrm{O}\left(\Delta t^{\,p+1}\right) \quad \text{for} \quad i, j = 1, 2 \tag{3.24} $$

If the proposed procedure is used, θ_2 can be stated in terms of θ_1 to satisfy the accuracy condition of Eq. (3.24) for any degrees of Hermite approximations. After selecting θ_2 in such a way as to satisfy the accuracy conditions given in Eq. (3.24), two eigenvalues of ${}^{a}\mathbf{A}$ can be expressed in a complex conjugate form. By setting $\xi = 0$ and $\omega\Delta t = \Omega$, two eigenvalues of ${}^{a}\mathbf{A}$ can be written in the form of

$$ \lambda_{1,2} = a(\theta_1, \theta_3, \Omega) \pm \sqrt{b(\theta_1, \theta_3, \Omega)} \tag{3.25} $$

where, a and b are proper real numbers. For the eigenvalues to remain complex conjugate, $b \leq 0$ should be provided for all $\Omega \geq 0$. In the proposed algorithms, this condition can be simply satisfied by selecting θ_3 to satisfy

$$ \lim_{\Omega \to \infty} b(\theta_1, \theta_3, \Omega) = 0 \tag{3.26} $$

From the condition given in Eq. (3.26), θ_3 can also be stated in terms of θ_1. At this point, the last remaining algorithmic parameter is θ_1. By selecting θ_1 judiciously, the new algorithms can achieve unconditional stability and dissipation control capability. To this end, θ_1 can be related to the spectral radius in the high frequency limit, which can be used for the control of algorithmic dissipation. The spectral radius of ${}^{a}\mathbf{A}$ is defined as

$$\rho({}^{a}\mathbf{A}) = \max\left(|\lambda_1|, |\lambda_2|\right) \tag{3.27}$$

where $|\lambda_1|$ and $|\lambda_2|$ are the absolute values of λ_1 and λ_2, respectively. By taking the limit of the spectral radius given in Eq. (3.27), the ultimate spectral radius is defined as

$$\mu = \lim_{\Omega\to\infty} \rho({}^{a}\mathbf{A}) \tag{3.28}$$

At last, θ_1 can be related to μ by using Eq. (3.28). μ can be chosen as any values varying from 0 to 1. If $\mu = 1$, the algorithms become non-dissipative, while $0 \leq \mu < 1$ will make algorithms dissipative. A user may chose proper values of μ depending on characteristics of analyses. If a user does not have any information about given problems, $\mu = 1$ is recommended as the standard case.

Here we present the weight parameters for the 3rd, 5th and 7th Hermite approximations. For the 3rd-degree Hermite approximation, the weight parameters can be chosen to achieve 3rd-order accuracy (4th-order accuracy is obtained if $\mu = 1$) as

$$\theta_1 = \frac{\mu+2}{3(\mu+1)}, \quad \theta_2 = \frac{2\mu^2+5\mu+5}{9(\mu+1)^2}, \quad \theta_3 = \frac{2\mu^3+6\mu^2+6\mu+4}{9(\mu+1)^3} \tag{3.29}$$

For the 5th-degree Hermite approximation, the weight parameters can be chosen to achieve 5th-order accuracy (6th-order accuracy is obtained if $\mu = 1$) as

$$\theta_1 = \frac{2\mu + 3}{5(\mu + 1)}, \quad \theta_2 = \frac{11\mu^2 + 28\mu + 21}{50(\mu + 1)^2}, \quad \theta_3 = \frac{36\mu^3 + 127\mu^2 + 158\mu + 79}{250(\mu + 1)^3} \tag{3.30}$$

For the 7th-degree Hermite approximation, the weight parameters can be chosen to achieve 7th-order accuracy (8th-order accuracy is obtained if $\mu = 1$) as

$$\theta_1 = \frac{3\mu + 4}{7(\mu + 1)}, \quad \theta_2 = \frac{11\mu^2 + 27\mu + 18}{49(\mu + 1)^2}, \quad \theta_3 = \frac{229\mu^3 + 804\mu^2 + 981\mu + 436}{1715(\mu + 1)^3} \tag{3.31}$$

As stated previously, only three independent weight parameters are required regardless the dgree of the approximation in the proposed procedure.

3.2.5 Final Form of Algorithms

After the optimization of the weight parameters, $\mathbf{u}_{s+1}, \dot{\mathbf{u}}_{s+1}, ..., \overset{(k-1)}{\mathbf{u}}_{s+1}$ can be stated in terms of $\mathbf{u}_s, \dot{\mathbf{u}}_s, ..., \overset{(k-1)}{\mathbf{u}}_s$ and other known properties by using Eqs. (3.9a) and (3.9b). The fully discrete equations (without elimination of second- and higher-order

nodal time derivatives of the displacement vector) can be expressed as

$$\begin{bmatrix} \mathbf{A}^{11} & \mathbf{A}^{12} & \mathbf{A}^{13} & \cdots & \mathbf{A}^{1k} \\ \mathbf{A}^{21} & \mathbf{A}^{22} & \mathbf{A}^{23} & \cdots & \mathbf{A}^{2k} \\ \mathbf{K} & \mathbf{C} & \mathbf{M} & & \\ & \ddots & \ddots & \ddots & \\ & & \mathbf{K} & \mathbf{C} & \mathbf{M} \end{bmatrix} \begin{Bmatrix} \mathbf{u}_{s+1} \\ \dot{\mathbf{u}}_{s+1} \\ \ddot{\mathbf{u}}_{s+1} \\ \vdots \\ \overset{(k-1)}{\mathbf{u}}_{s+1} \end{Bmatrix} = \begin{bmatrix} \mathbf{B}^{11} & \mathbf{B}^{12} & \mathbf{B}^{13} & \cdots & \mathbf{B}^{1k} \\ \mathbf{B}^{21} & \mathbf{B}^{22} & \mathbf{B}^{23} & \cdots & \mathbf{B}^{2k} \\ & & & & \\ & & & & \\ & & & & \end{bmatrix} \begin{Bmatrix} \mathbf{u}_{s} \\ \dot{\mathbf{u}}_{s} \\ \ddot{\mathbf{u}}_{s} \\ \vdots \\ \overset{(k-1)}{\mathbf{u}}_{s} \end{Bmatrix} + \begin{Bmatrix} \mathbf{c}^1 \\ \mathbf{c}^2 \\ \overset{(0)}{\mathbf{f}}_{s+1} \\ \vdots \\ \overset{(k-3)}{\mathbf{f}}_{s+1} \end{Bmatrix} \tag{3.32}$$

where $k = \frac{p+1}{2}$, $\mathbf{A}^{ij}$ and $\mathbf{B}^{ij}$ are matrices which can be defined in terms of $\mathbf{M}$, $\mathbf{C}$, $\mathbf{K}$, μ, and Δt; $\mathbf{c}^1$ and $\mathbf{c}^2$ are vectors which can be defined in terms of $\mathbf{f}(t)$, Δt, and μ. By using Eq. (3.11), second- and higher-order nodal time derivatives at $t = t_{s+1}$ in Eq. (3.32) can be stated in terms of $\mathbf{u}_{s+1}$ and $\dot{\mathbf{u}}_{s+1}$ as

$$\begin{Bmatrix} \overset{(2)}{\mathbf{u}}_{s+1} \\ \vdots \\ \overset{(k-1)}{\mathbf{u}}_{s+1} \end{Bmatrix} = -\left[\mathbf{\Lambda}\right]^{-1}\left[\mathbf{\Delta}\right] \begin{Bmatrix} \mathbf{u}_{s+1} \\ \dot{\mathbf{u}}_{s+1} \end{Bmatrix} + \left[\mathbf{\Lambda}\right]^{-1} \begin{Bmatrix} \overset{(0)}{\mathbf{f}}_{s+1} \\ \vdots \\ \overset{(k-3)}{\mathbf{f}}_{s+1} \end{Bmatrix} \tag{3.33}$$

where $\overset{(2)}{\mathbf{u}}_{s+1} = \ddot{\mathbf{u}}_{s+1}$, and matrices $[\mathbf{\Lambda}]$ and $[\mathbf{\Delta}]$ are defined as

$$[\mathbf{\Lambda}] = \begin{bmatrix} \mathbf{M} & & & & \\ \mathbf{C} & \mathbf{M} & & & \\ \mathbf{K} & \mathbf{C} & \mathbf{M} & & \\ & \ddots & \ddots & \ddots & \\ & & \mathbf{K} & \mathbf{C} & \mathbf{M} \end{bmatrix}, \quad [\mathbf{\Delta}] = \begin{bmatrix} \mathbf{K} & \mathbf{C} \\ \mathbf{0} & \mathbf{K} \\ \vdots & \mathbf{0} \\ & \vdots \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{3.34}$$

Similarly, by using Eq. (3.12), second- and higher-order nodal time derivatives at $t = t_s$ in Eq. (3.32) can also be stated in terms of $\mathbf{u}_s$ and $\dot{\mathbf{u}}_s$ as

$$\begin{Bmatrix} \overset{(2)}{\mathbf{u}}_s \\ \vdots \\ \overset{(k-1)}{\mathbf{u}}_s \end{Bmatrix} = -[\mathbf{\Lambda}]^{-1}[\mathbf{\Delta}] \begin{Bmatrix} \mathbf{u}_s \\ \dot{\mathbf{u}}_s \end{Bmatrix} + [\mathbf{\Lambda}]^{-1} \begin{Bmatrix} \overset{(0)}{\mathbf{f}}_s \\ \vdots \\ \overset{(k-3)}{\mathbf{f}}_s \end{Bmatrix} \tag{3.35}$$

Now Eqs. (3.33) and (3.35) can be used to eliminate the second- and higher-order nodal time derivatives in Eq. (3.32). By substituting Eqs. (3.33) and (3.35) into Eq. (3.32), condensed form of algorithm can be obtained as

$$\begin{aligned} &\left(\begin{bmatrix} \mathbf{A}^{11} & \mathbf{A}^{12} \\ \mathbf{A}^{21} & \mathbf{A}^{22} \end{bmatrix} - [\mathbf{\Gamma}_{\mathrm{A}}][\mathbf{\Lambda}]^{-1}[\mathbf{\Delta}] \right) \begin{Bmatrix} \mathbf{u}_{s+1} \\ \dot{\mathbf{u}}_{s+1} \end{Bmatrix} \\ &= \left(\begin{bmatrix} \mathbf{B}^{11} & \mathbf{B}^{12} \\ \mathbf{B}^{21} & \mathbf{B}^{22} \end{bmatrix} - [\mathbf{\Gamma}_{\mathrm{B}}][\mathbf{\Lambda}]^{-1}[\mathbf{\Delta}] \right) \begin{Bmatrix} \mathbf{u}_s \\ \dot{\mathbf{u}}_s \end{Bmatrix} + [\mathbf{\Gamma}_{\mathrm{B}}][\mathbf{\Lambda}]^{-1} \begin{Bmatrix} \overset{(0)}{\mathbf{f}}_s \\ \vdots \\ \overset{(k-3)}{\mathbf{f}}_s \end{Bmatrix} \\ &\quad + \begin{Bmatrix} \mathbf{c}^1 \\ \mathbf{c}^2 \end{Bmatrix} - [\mathbf{\Gamma}_{\mathrm{A}}][\mathbf{\Lambda}]^{-1} \begin{Bmatrix} \overset{(0)}{\mathbf{f}}_{s+1} \\ \vdots \\ \overset{(k-3)}{\mathbf{f}}_{s+1} \end{Bmatrix} \end{aligned} \tag{3.36}$$

where matrices $[\mathbf{\Gamma}_{\mathrm{A}}]$ and $[\mathbf{\Gamma}_{\mathrm{B}}]$ are defined as

$$[\mathbf{\Gamma}_{\mathrm{A}}] = \begin{bmatrix} \mathbf{A}^{13} & \cdots & \mathbf{A}^{1k} \\ \mathbf{A}^{23} & \cdots & \mathbf{A}^{2k} \end{bmatrix}, \quad [\mathbf{\Gamma}_{\mathrm{B}}] = \begin{bmatrix} \mathbf{B}^{13} & \cdots & \mathbf{B}^{1k} \\ \mathbf{B}^{23} & \cdots & \mathbf{B}^{2k} \end{bmatrix} \tag{3.37}$$

In a practical view point, rewriting Eq. (3.32) as the condensed form given in Eq. (3.36) can be very beneficial, because any algorithms developed by using the proposed procedure can be solve in the same way that the $2m \times 2m$ system is solved, if the coefficient matrix of $\mathbf{u}_{s+1}, \dot{\mathbf{u}}_{s+1}$ in Eq. (3.36) is properly constructed. In the new algorithms, the coefficient matrix of $\mathbf{u}_{s+1}, \dot{\mathbf{u}}_{s+1}$ presented in Eq. (3.36) can be constructed without noticeable increase of computational effort, due to the unique computational structure of $[\mathbf{\Lambda}]$ as presented in Eq. (3.34). If $\mathbf{M}$ is a diagonal matrix, $[\mathbf{\Lambda}]$ given in Eq. (3.34) automatically becomes lower triangular form [94], then the matrix operations of $[\mathbf{\Gamma}_{\mathrm{A}}][\mathbf{\Lambda}]^{-1}[\mathbf{\Delta}]$, $[\mathbf{\Gamma}_{\mathrm{B}}][\mathbf{\Lambda}]^{-1}[\mathbf{\Delta}]$, $[\mathbf{\Gamma}_{\mathrm{A}}][\mathbf{\Lambda}]^{-1}$ and $[\mathbf{\Gamma}_{\mathrm{B}}][\mathbf{\Lambda}]^{-1}$ in Eq. (3.36) can be done very efficiently. Here we present 3rd-, 5th-, and 7th-order algorithms from the current procedure. The 3rd-order algorithm can be written as

$$\begin{bmatrix} \mathbf{A}^{11} & \mathbf{A}^{12} \\ \mathbf{A}^{21} & \mathbf{A}^{22} \end{bmatrix} \begin{Bmatrix} \mathbf{u}_{s+1} \\ \dot{\mathbf{u}}_{s+1} \end{Bmatrix} = \begin{bmatrix} \mathbf{B}^{11} & \mathbf{B}^{12} \\ \mathbf{B}^{21} & \mathbf{B}^{22} \end{bmatrix} \begin{Bmatrix} \mathbf{u}_{s} \\ \dot{\mathbf{u}}_{s} \end{Bmatrix} = \begin{Bmatrix} \mathbf{c}^{1} \\ \mathbf{c}^{2} \end{Bmatrix} \tag{3.38}$$

where,

$$\mathbf{A}^{11} = \frac{2(\mu-1)}{(\mu+1)(\Delta t)^2}\mathbf{M} + \frac{2(\mu^2+4\mu+1)}{3(\mu+1)^2\Delta t}\mathbf{C} + \frac{(2\mu+1)(\mu^2+4\mu+7)}{9(\mu+1)^3}\mathbf{K}$$

$$\mathbf{A}^{12} = \frac{2}{(\mu+1)\Delta t}\mathbf{M} - \frac{\mu-1}{3(\mu+1)^2}\mathbf{C} - \frac{(\mu^2+4\mu+1)\Delta t}{9(\mu+1)^3}\mathbf{K}$$

$$\mathbf{A}^{21} = -\frac{12}{(\Delta t)^2}\mathbf{M} + \frac{2(\mu-1)}{(\mu+1)\Delta t}\mathbf{C} + \frac{2(\mu^2+4\mu+1)}{3(\mu+1)^2}\mathbf{K}$$

$$\mathbf{A}^{22} = \frac{6}{\Delta t}\mathbf{M} + \frac{2}{(\mu+1)}\mathbf{C} - \frac{(\mu-1)\Delta t}{3(\mu+1)^2}\mathbf{K}$$

$$\mathbf{B}^{11} = \frac{2(\mu-1)}{(\mu+1)(\Delta t)^2}\mathbf{M} + \frac{2(\mu^2+4\mu+1)}{3(\mu+1)^2\Delta t}\mathbf{C} - \frac{(\mu+2)(7\mu^2+4\mu+1)}{9(\mu+1)^3}\mathbf{K}$$

$$\mathbf{B}^{12} = \frac{2\mu}{(\mu+1)\Delta t}\mathbf{M} - \frac{\mu(\mu-1)}{3(\mu+1)^2}\mathbf{C} - \frac{\mu(\mu^2+4\mu+1)\Delta t}{9(\mu+1)^3}\mathbf{K}$$

$$\mathbf{B}^{21} = -\frac{12}{(\Delta t)^2}\mathbf{M} + \frac{2(\mu-1)}{(\mu+1)\Delta t}\mathbf{C} + \frac{2(\mu^2+4\mu+1)}{3(\mu+1)^2}\mathbf{K}$$

$$\mathbf{B}^{22} = -\frac{6}{\Delta t}\mathbf{M} + \frac{2\mu}{(\mu+1)}\mathbf{C} - \frac{\mu(\mu-1)\Delta t}{3(\mu+1)^2}\mathbf{K}$$

$$\begin{aligned}\mathbf{c}^1 =& \frac{(\mu+2)(7\mu^2+4\mu+1)}{9(\mu+1)^3}\mathbf{f}_s + \frac{(\mu^2+4\mu+1)\Delta t}{9(\mu+1)^3}\dot{\mathbf{f}}_s \\ &+ \frac{(2\mu+1)(\mu^2+4\mu+7)}{9(\mu+1)^3}\mathbf{f}_{s+1} - \frac{(\mu^2+4\mu+1)\Delta t}{9(\mu+1)^3}\dot{\mathbf{f}}_{s+1}\end{aligned}$$

$$\mathbf{c}^2 = -\frac{2(\mu^2+4\mu+1)}{3(\mu+1)^2}\mathbf{f}_s + \frac{\mu(\mu-1)\Delta t}{3(\mu+1)^2}\dot{\mathbf{f}}_s + \frac{2(\mu^2+4\mu+1)}{3(\mu+1)^2}\mathbf{f}_{s+1} - \frac{(\mu-1)\Delta t}{3(\mu+1)^2}\dot{\mathbf{f}}_{s+1}$$

The 5th-order algorithm can be written as

$$\begin{aligned}&\left(\begin{bmatrix}\mathbf{A}^{11} & \mathbf{A}^{12}\\ \mathbf{A}^{21} & \mathbf{A}^{22}\end{bmatrix} - \begin{bmatrix}\mathbf{A}^{13}\\ \mathbf{A}^{23}\end{bmatrix}[\mathbf{M}]^{-1}[\mathbf{K} \quad \mathbf{C}]\right)\begin{Bmatrix}\mathbf{u_{s+1}}\\ \dot{\mathbf{u}}_{s+1}\end{Bmatrix} \\ &= \left(\begin{bmatrix}\mathbf{B}^{11} & \mathbf{B}^{12}\\ \mathbf{B}^{21} & \mathbf{B}^{22}\end{bmatrix} - \begin{bmatrix}\mathbf{B}^{13}\\ \mathbf{B}^{23}\end{bmatrix}[\mathbf{M}]^{-1}[\mathbf{K} \quad \mathbf{C}]\right)\begin{Bmatrix}\mathbf{u_s}\\ \dot{\mathbf{u}}_s\end{Bmatrix} + \begin{bmatrix}\mathbf{B}^{13}\\ \mathbf{B}^{23}\end{bmatrix}[\mathbf{M}]^{-1}\begin{Bmatrix}\mathbf{f}_s\\ \dot{\mathbf{f}}_s\end{Bmatrix} \\ &\quad - \begin{bmatrix}\mathbf{A}^{13}\\ \mathbf{A}^{23}\end{bmatrix}[\mathbf{M}]^{-1}\begin{Bmatrix}\mathbf{f}_{s+1}\\ \dot{\mathbf{f}}_{s+1}\end{Bmatrix} + \begin{Bmatrix}\mathbf{c}^1\\ \mathbf{c}^2\end{Bmatrix}\end{aligned} \tag{3.39}$$

where,

$$\mathbf{A}^{11} = -\frac{72(\mu-1)}{(\mu+1)(\Delta t)^2}\mathbf{M} - \frac{24(\mu^2+8\mu+1)}{5(\mu+1)^2\Delta t}\mathbf{C} + \frac{6(\mu-1)(7\mu^2+26\mu+7)}{25(\mu+1)^3}\mathbf{K}$$

$$\mathbf{A}^{12} = \frac{24(\mu-2)}{(\mu+1)\Delta t}\mathbf{M} + \frac{6(3\mu^2+6\mu+1)}{5(\mu+1)^2}\mathbf{C} + \frac{6(\mu^3+6\mu^2+25\mu+8)\Delta t}{25(\mu+1)^3}\mathbf{K}$$

$$\mathbf{A}^{13} = \frac{12}{\mu+1}\mathbf{M} - \frac{6(\mu-1)\Delta t}{5(\mu+1)^2}\mathbf{C} - \frac{2(\mu^2+8\mu+1)(\Delta t)^2}{25(\mu+1)^3}\mathbf{K}$$

$$\mathbf{A}^{21} = \frac{720}{(\Delta t)^2}\mathbf{M} - \frac{72(\mu-1)}{(\mu+1)\Delta t}\mathbf{C} - \frac{24(\mu^2+8\mu+1)}{5(\mu+1)^2}\mathbf{K}$$

$$\mathbf{A}^{22} = -\frac{360}{\Delta t}\mathbf{M} + \frac{24(\mu-2)}{\mu+1}\mathbf{C} + \frac{6(3\mu^2+16\mu+1)\Delta t}{5(\mu+1)^2}\mathbf{K}$$

$$\mathbf{A}^{23} = 60\mathbf{M} + \frac{12\Delta t}{\mu+1}\mathbf{C} - \frac{6(\mu-1)(\Delta t)^2}{5(\mu+1)^2}\mathbf{K}$$

$$\mathbf{B}^{11} = -\frac{72(\mu-1)}{(\mu+1)(\Delta t)^2}\mathbf{M} - \frac{24(\mu^2+8\mu+1)}{5(\mu+1)^2\Delta t}\mathbf{C} + \frac{6(\mu-1)(7\mu^2+26\mu+7)}{25(\mu+1)^3}\mathbf{K}$$

$$\mathbf{B}^{12} = -\frac{24(2\mu-1)}{(\mu+1)\Delta t}\mathbf{M} - \frac{6(\mu^2+16\mu+3)}{5(\mu+1)^2}\mathbf{C} - \frac{6(8\mu^3+25\mu^2+6\mu+1)\Delta t}{25(\mu+1)^3}\mathbf{K}$$

$$\mathbf{B}^{13} = -\frac{12\mu}{\mu+1}\mathbf{M} + \frac{6\mu(\mu-1)\Delta t}{5(\mu+1)^2}\mathbf{C} + \frac{2\mu(\mu^2+8\mu+1)(\Delta t)^2}{25(\mu+1)^3}\mathbf{K}$$

$$\mathbf{B}^{21} = \frac{720}{(\Delta t)^2}\mathbf{M} - \frac{72(\mu-1)}{(\mu+1)\Delta t}\mathbf{C} - \frac{24(\mu^2+8\mu+1)}{5(\mu+1)^2}\mathbf{K}$$

$$\mathbf{B}^{22} = \frac{360}{\Delta t}\mathbf{M} - \frac{24(2\mu-1)}{\mu+1}\mathbf{C} - \frac{6(\mu^2+16\mu+3)\Delta t}{5(\mu+1)^2}\mathbf{K}$$

$$\mathbf{B}^{23} = 60\mathbf{M} - \frac{12\mu\Delta t}{\mu+1}\mathbf{C} + \frac{6\mu(\mu-1)(\Delta t)^2}{5(\mu+1)^2}\mathbf{K}$$

$$\mathbf{c}^1 = -\frac{6(\mu-1)(7\mu^2+26\mu+7)}{25(\mu+1)^3}\mathbf{f}_s - \frac{6(8\mu^3+25\mu^2+6\mu+1)\Delta t}{25(\mu+1)^3}\dot{\mathbf{f}}_s - \frac{2(\mu^2+8\mu+1)(\Delta t)^2}{25(\mu+1)^3}\ddot{\mathbf{f}}_s$$
$$+\frac{6(\mu-1)(7\mu^2+26\mu+7)}{25(\mu+1)^3}\mathbf{f}_{s+1} + \frac{6(\mu^3+6\mu^2+25\mu+8)\Delta t}{25(\mu+1)^3}\dot{\mathbf{f}}_{s+1} - \frac{2(\mu^2+8\mu+1)(\Delta t)^2}{25(\mu+1)^3}\ddot{\mathbf{f}}_{s+1}$$

$$\mathbf{c}^2 = \frac{24(\mu^2+8\mu+1)}{5(\mu+1)^2}\mathbf{f}_s + \frac{6(\mu^2+16\mu+3)\Delta t}{5(\mu+1)^2}\dot{\mathbf{f}}_s - \frac{6(\mu-1)(\Delta t)^2}{5(\mu+1)^2}\ddot{\mathbf{f}}_s$$
$$-\frac{24(\mu^2+8\mu+1)}{5(\mu+1)^2}\mathbf{f}_{s+1} + \frac{6(3\mu^2+16\mu+1)\Delta t}{5(\mu+1)^2}\dot{\mathbf{f}}_{s+1} - \frac{6(\mu-1)(\Delta t)^2}{5(\mu+1)^2}\ddot{\mathbf{f}}_{s+1}$$

The 7th-order algorithm can be written as

$$
\begin{aligned}
&\left(\begin{bmatrix} \mathbf{A}^{11} & \mathbf{A}^{12} \\ \mathbf{A}^{21} & \mathbf{A}^{22} \end{bmatrix} - \begin{bmatrix} \mathbf{A}^{13} & \mathbf{A}^{14} \\ \mathbf{A}^{23} & \mathbf{A}^{24} \end{bmatrix} \begin{bmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{C} & \mathbf{M} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{K} & \mathbf{C} \\ \mathbf{0} & \mathbf{K} \end{bmatrix} \right) \begin{Bmatrix} \mathbf{u_{s+1}} \\ \dot{\mathbf{u}}_{s+1} \end{Bmatrix} \\
&= \left(\begin{bmatrix} \mathbf{B}^{11} & \mathbf{B}^{12} \\ \mathbf{B}^{21} & \mathbf{B}^{22} \end{bmatrix} - \begin{bmatrix} \mathbf{B}^{13} & \mathbf{B}^{14} \\ \mathbf{B}^{23} & \mathbf{B}^{24} \end{bmatrix} \begin{bmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{C} & \mathbf{M} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{K} & \mathbf{C} \\ \mathbf{0} & \mathbf{K} \end{bmatrix} \right) \begin{Bmatrix} \mathbf{u_s} \\ \dot{\mathbf{u}}_s \end{Bmatrix} \\
&\quad + \begin{bmatrix} \mathbf{B}^{13} & \mathbf{B}^{14} \\ \mathbf{B}^{23} & \mathbf{B}^{24} \end{bmatrix} \begin{bmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{C} & \mathbf{M} \end{bmatrix}^{-1} \begin{Bmatrix} \mathbf{f}_s \\ \dot{\mathbf{f}}_s \end{Bmatrix} \\
&\quad - \begin{bmatrix} \mathbf{A}^{13} & \mathbf{A}^{14} \\ \mathbf{A}^{23} & \mathbf{A}^{24} \end{bmatrix} \begin{bmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{C} & \mathbf{M} \end{bmatrix}^{-1} \begin{Bmatrix} \mathbf{f}_{s+1} \\ \dot{\mathbf{f}}_{s+1} \end{Bmatrix} + \begin{Bmatrix} \mathbf{c}^1 \\ \mathbf{c}^2 \end{Bmatrix}
\end{aligned}
\tag{3.40}
$$

where

$$\mathbf{A}^{11} = \frac{7200(\mu-1)}{(\mu+1)(\Delta t)^2}\mathbf{M} + \frac{1440(\mu^2+12\mu+1)}{7(\mu+1)^2\Delta t}\mathbf{C} - \frac{360(\mu-1)(9\mu^2+38\mu+9)}{49(\mu+1)^3}\mathbf{K}$$

$$\mathbf{A}^{12} = -\frac{1440(2\mu-3)}{(\mu+1)\Delta t}\mathbf{M} - \frac{360(3\mu^2+24\mu+1)}{7(\mu+1)^2}\mathbf{C} + \frac{120(8\mu^3+21\mu^2-66\mu-19)\Delta t}{49(\mu+1)^3}\mathbf{K}$$

$$\mathbf{A}^{13} = \frac{360(\mu-3)}{\mu+1}\mathbf{M} + \frac{120(2\mu^2+13\mu-1)\Delta t}{7(\mu+1)^2}\mathbf{C} + \frac{12(4\mu^3+29\mu^2+196\mu+51)(\Delta t)^2}{49(\mu+1)^3}\mathbf{K}$$

$$\mathbf{A}^{14} = \frac{120\Delta t}{\mu+1}\mathbf{M} - \frac{60(\mu-1)(\Delta t)^2}{7(\mu+1)^2}\mathbf{C} - \frac{12(\mu^2+12\mu+1)(\Delta t)^3}{49(\mu+1)^3}\mathbf{K}$$

$$\mathbf{A}^{21} = -\frac{100800}{(\Delta t)^2}\mathbf{M} + \frac{7200(\mu-1)}{(\mu+1)\Delta t}\mathbf{C} + \frac{1440(\mu^2+12\mu+1)}{7(\mu+1)^2}\mathbf{K}$$

$$\mathbf{A}^{22} = \frac{50400}{\Delta t}\mathbf{M} - \frac{1440(2\mu-3)}{\mu+1}\mathbf{C} + \frac{360(3\mu^2+24\mu+1)\Delta t}{7(\mu+1)^2}\mathbf{K}$$

$$\mathbf{A}^{23} = -10080\mathbf{M} + \frac{360(\mu-3)\Delta t}{\mu+1}\mathbf{C} + \frac{120(2\mu^2+13\mu-1)(\Delta t)^2}{7(\mu+1)^2}\mathbf{K}$$

$$\mathbf{A}^{24} = 840\Delta t\mathbf{M} + \frac{120(\Delta t)^2}{\mu+1}\mathbf{C} - \frac{60(\mu-1)(\Delta t)^3}{7(\mu+1)^2}\mathbf{K}$$

$$\mathbf{B}^{11} = \frac{7200(\mu-1)}{(\mu+1)(\Delta t)^2}\mathbf{M} + \frac{1440(\mu^2+12\mu+1)}{7(\mu+1)^2\Delta t}\mathbf{C} - \frac{360(\mu-1)(9\mu^2+38\mu+9)}{49(\mu+1)^3}\mathbf{K}$$

$$\mathbf{B}^{12} = \frac{1440(3\mu-2)}{(\mu+1)\Delta t}\mathbf{M} + \frac{360(\mu^2+24\mu+3)}{7(\mu+1)^2}\mathbf{C} - \frac{120(19\mu^3+66\mu^2-21\mu-8)\Delta t}{49(\mu+1)^3}\mathbf{K}$$

$$\mathbf{B}^{13} = \frac{360(3\mu-1)}{\mu+1}\mathbf{M} - \frac{120(\mu^2-13\mu-2)\Delta t}{7(\mu+1)^2}\mathbf{C} - \frac{12(51\mu^3+196\mu^2+26\mu+4)(\Delta t)^2}{49(\mu+1)^3}\mathbf{K}$$

$$\mathbf{B}^{14} = \frac{120\Delta t}{\mu+1}\mathbf{M} - \frac{60(\mu-1)(\Delta t)^2}{7(\mu+1)^2}\mathbf{C} - \frac{12(\mu^2+12\mu+1)(\Delta t)^3}{49(\mu+1)^3}\mathbf{K}$$

$$\mathbf{B}^{21} = -\frac{100800}{(\Delta t)^2}\mathbf{M} + \frac{7200(\mu-1)}{(\mu+1)\Delta t}\mathbf{C} + \frac{1440(\mu^2+12\mu+1)}{7(\mu+1)^2}\mathbf{K}$$

$$\mathbf{B}^{22} = -\frac{50400}{\Delta t}\mathbf{M} + \frac{1440(3\mu-2)}{\mu+1}\mathbf{C} + \frac{360(\mu^2+24\mu+3)\Delta t}{7(\mu+1)^2}\mathbf{K}$$

$$\mathbf{B}^{23} = -10080\mathbf{M} + \frac{360(3\mu-1)\Delta t}{\mu+1}\mathbf{C} - \frac{120(\mu^2-13\mu-2)(\Delta t)^2}{7(\mu+1)^2}\mathbf{K}$$

$$\mathbf{B}^{24} = -840\Delta t\mathbf{M} + \frac{120(\Delta t)^2}{\mu+1}\mathbf{C} - \frac{60(\mu-1)(\Delta t)^3}{7(\mu+1)^2}\mathbf{K}$$

$$\begin{aligned}\mathbf{c}^1 = {} & \frac{360(\mu-1)(9\mu^2+38\mu+9)}{49(\mu+1)^3}\mathbf{f}_s + \frac{120(19\mu^3+66\mu^2-21\mu-8)\Delta t}{49(\mu+1)^3}\dot{\mathbf{f}}_s \\ & + \frac{12(51\mu^3+196\mu^2+29\mu+4)(\Delta t)^2}{49(\mu+1)^3}\ddot{\mathbf{f}}_s + \frac{12(\mu^2+12\mu+1)(\Delta t)^3}{49(\mu+1)^3}\dddot{\mathbf{f}}_s \\ & - \frac{360(\mu-1)(9\mu^2+38\mu+9)}{49(\mu+1)^3}\mathbf{f}_{s+1} + \frac{120(8\mu^3+21\mu^2-66\mu-19)\Delta t}{49(\mu+1)^3}\dot{\mathbf{f}}_{s+1} \\ & + \frac{12(4\mu^3+29\mu^2+196\mu+51)(\Delta t)^2}{49(\mu+1)^3}\ddot{\mathbf{f}}_{s+1} - \frac{12(\mu^2+12\mu+1)(\Delta t)^3}{49(\mu+1)^3}\dddot{\mathbf{f}}_{s+1}\end{aligned}$$

$$\begin{aligned}\mathbf{c}^2 = {} & -\frac{1440(\mu^2+12\mu+1)}{7(\mu+1)^2}\mathbf{f}_s - \frac{360(\mu^2+24\mu+3)\Delta t}{7(\mu+1)^2}\dot{\mathbf{f}}_s \\ & + \frac{120(\mu^2-13\mu-2)(\Delta t)^2}{7(\mu+1)^2}\ddot{\mathbf{f}}_s + \frac{60\mu(\mu-1)(\Delta t)^3}{7(\mu+1)^2}\dddot{\mathbf{f}}_s \\ & + \frac{1440(\mu^2+12\mu+1)}{7(\mu+1)^2}\mathbf{f}_{s+1} - \frac{360(3\mu^2+24\mu+1)\Delta t}{7(\mu+1)^2}\dot{\mathbf{f}}_{s+1} \\ & + \frac{120(2\mu^2+13\mu-1)(\Delta t)^2}{7(\mu+1)^2}\ddot{\mathbf{f}}_s - \frac{60(\mu-1)(\Delta t)^3}{7(\mu+1)^2}\dddot{\mathbf{f}}_{s+1}\end{aligned}$$

3.2.6 Nonlinear Analysis

To apply the new algorithms to nonlinear analysis, we can consider the semi-discrete form of nonlinear dynamic system which can be written as

$$\mathbf{M}\ddot{\mathbf{u}}(t) + \mathbf{C}\dot{\mathbf{u}}(t) + \mathbf{n}(t) = \mathbf{f}(t) \tag{3.41}$$

where, $\mathbf{n}(t)$ is the nonlinear internal force vector. To apply the new algorithms to the semi-discrete governing equation given in Eq. (3.41), it should be rewritten in a proper form. The simplest admissible modification of Eq. (3.41) can be done by decomposing $\mathbf{n}(t)$ into the linear and nonlinear parts. In many cases, $\mathbf{n}(t)$ can be decomposed as $\mathbf{Ku}(t) + \hat{\mathbf{n}}(t)$, where $\mathbf{Ku}(t)$ is the linear part of $\mathbf{n}(t)$, and $\hat{\mathbf{n}}(t)$ is the nonlinear part of $\mathbf{n}(t)$. If $\mathbf{n}(t) = \mathbf{Ku}(t) + \hat{\mathbf{n}}(t)$ is used, Eq. (3.41) can be rewritten as

$$\mathbf{M}\ddot{\mathbf{u}}(t) + \mathbf{C}\dot{\mathbf{u}}(t) + \mathbf{Ku}(t) = \hat{\mathbf{f}}(t) \tag{3.42}$$

where, $\hat{\mathbf{f}}(t) = \mathbf{f}(t) - \hat{\mathbf{n}}(t)$. Then, $\mathbf{f}_{s+1}, \dot{\mathbf{f}}_{s+1}, ..., \overset{(k-1)}{\mathbf{f}}_{s+1}$ and $\mathbf{f}_s, \dot{\mathbf{f}}_s, ..., \overset{(k-1)}{\mathbf{f}}_s$ in Eq. (3.36) can be replaced by $\hat{\mathbf{f}}_{s+1}, \dot{\hat{\mathbf{f}}}_{s+1}, ..., \overset{(k-1)}{\hat{\mathbf{f}}}_{s+1}$ and $\hat{\mathbf{f}}_s, \dot{\hat{\mathbf{f}}}_s, ..., \overset{(k-1)}{\hat{\mathbf{f}}}_s$ for the nonlinear analysis. After replacing the force terms, Eq. (3.36) can be solved by using the direct iterative nonlinear equation solving method. Related discussions can be found in Ref. [95]. However, the decomposition of $\mathbf{n}(t)$ may not be possible in some special cases, and this type of nonlinear analysis can experience poor convergence of nonlinear solutions, if the nonlinearity is sever. Then one should reduce the size of Δt to get proper convergence of nonlinear solutions.

3.3 Analysis

3.3.1 Stability and Algorithmic Dissipation Control

As discussed previously, we can use the single-degree-of-freedom problem to check accuracy and stability of the new algorithms. The algorithmic amplification matrix can be obtained by applying a time integration algorithm to the single-degree-of-freedom problem, and the spectral radius can also be obtained from the algorithmic amplification matrix. The spectral radius is defined as the maximum absolute value of the eigenvalues of $\rho({}^{a}\mathbf{A})$, and it can be used as the measure of stability and dissipation of the algorithm. Thus, we can check stability of the new algorithms by investigating the variation of the spectral radius for varying values of $\frac{\Delta t}{T}$, T being the period of the single-degree-of-freedom problem. An algorithm is said to be unconditionally stable if $0 \leq \rho({}^{a}\mathbf{A}) \leq 1$ is provided. Figure 3.5 shows that the current algorithms are unconditionally stable if μ is chosen in the range of $0 \leq \mu \leq 1$.

As presented in Figure 3.5, the algorithmic dissipation in the high frequency limit can also be adjusted as a desired level through the proper specification of μ. In many practical cases, algorithmic dissipation can be used to filter out spurious high frequency responds caused due to poor representations of the spatial domain of original governing PDEs. In the current algorithms, algorithmic dissipations can be effectively controlled through the specification of μ.

3.3.2 Accuracy

We already imposed desired order of accuracies on algorithms through the optimization of the weight parameters by comparing the entries of ${}^{e}\mathbf{A}$ and ${}^{a}\mathbf{A}$. The exact and numerical amplification matrices in Eq. (3.24) were obtained by using the exact and algorithmic discrete solutions stated in terms of the initial conditions. To define the order of accuracy of an algorithm at any arbitrary time steps, however, a

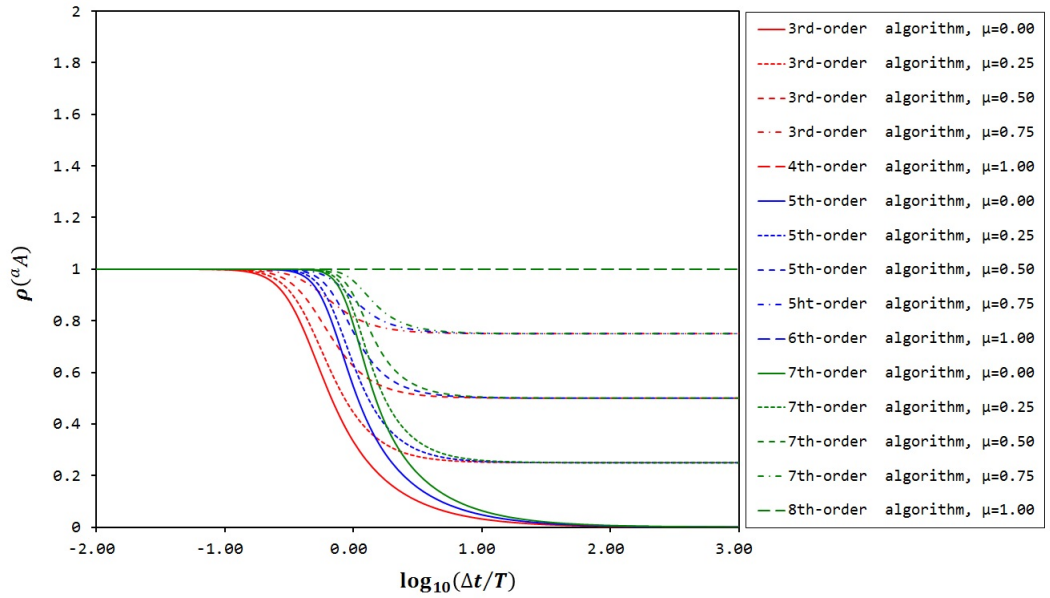

Figure 3.5: Spectral radii versus $\Delta t/T$ for current 3rd-, 5th-, and 7th-order algorithms with varying values of μ.

proper local error should be defined. In literatures [5, 17, 49], the order of accuracy of time integration algorithms has been defined between arbitrary sth and $(s+1)$th time step solutions by utilizing the local truncation error. The local truncation error of algorithms can be defined as

$$\tau_e(t_s) = \frac{1}{\Delta t^2}\Big(u(t_s+\Delta t) - 2A_1 u(t_s) + A_2 u(t_s-\Delta t)\Big) \tag{3.43}$$

where $A_1 = 1/2\ \mathrm{tr}({}^a\mathbf{A})$, $A_2 = \det({}^a\mathbf{A})$, and $u(t)$ is the exact solution of the single-degree-of-problem given in Eq. (3.17) for the case of $f(t) = 0$. From Eq. (3.43), the order of accuracy of an algorithm is defined as kth order if $\tau_e = \mathrm{O}\left(\Delta t^k\right)$ is provided. Again, we note that the order of accuracy of current algorithms obtained from the pth-degree Hermite interpolation functions is $(p-1)$th-order (for $0 \leq \mu < 1$) or pth-order (for $\mu = 1$). Since the pth-degree Hermite approximation includes $\frac{p+1}{2}$ unknowns, we also define the order of accuracy as $(2n-1)$th- and $(2n)$th-order accurate, where $n = \frac{p+1}{2}$.

The order of accuracy is a mathematically important information of an algorithm, but it is not a practical measure in most of analyses. Sometimes, the accuracy of an algorithm is often explained by using more practical measures, such as the relative period error and the algorithmic damping ratio. The relative period error and the algorithmic damping ratio can be computed by using the exact solution and the invariants of the numerical amplification matrix. The relative period error is defined as $\left(\bar{T} - T\right)/T$ where, the exact period is $T = 2\ \pi/\omega$ and the numerically obtained period is $\bar{T} = 2\pi/\bar{\omega}$. The algorithmic damping ratio is defined as $\bar{\xi} = -\ln(A_2)/(2\ \bar{\Omega})$. Here $\bar{\omega} = \mathrm{acrtan}\left(\sqrt{A_2/A_1^2 - 1}\right)/\left(\Delta t\sqrt{1-\xi^2}\right)$ and $\bar{\Omega} = \bar{\omega}\ \Delta t$. The relative period error and the algorithmic damping ratio are presented in Figs. 3.6 and 3.7.

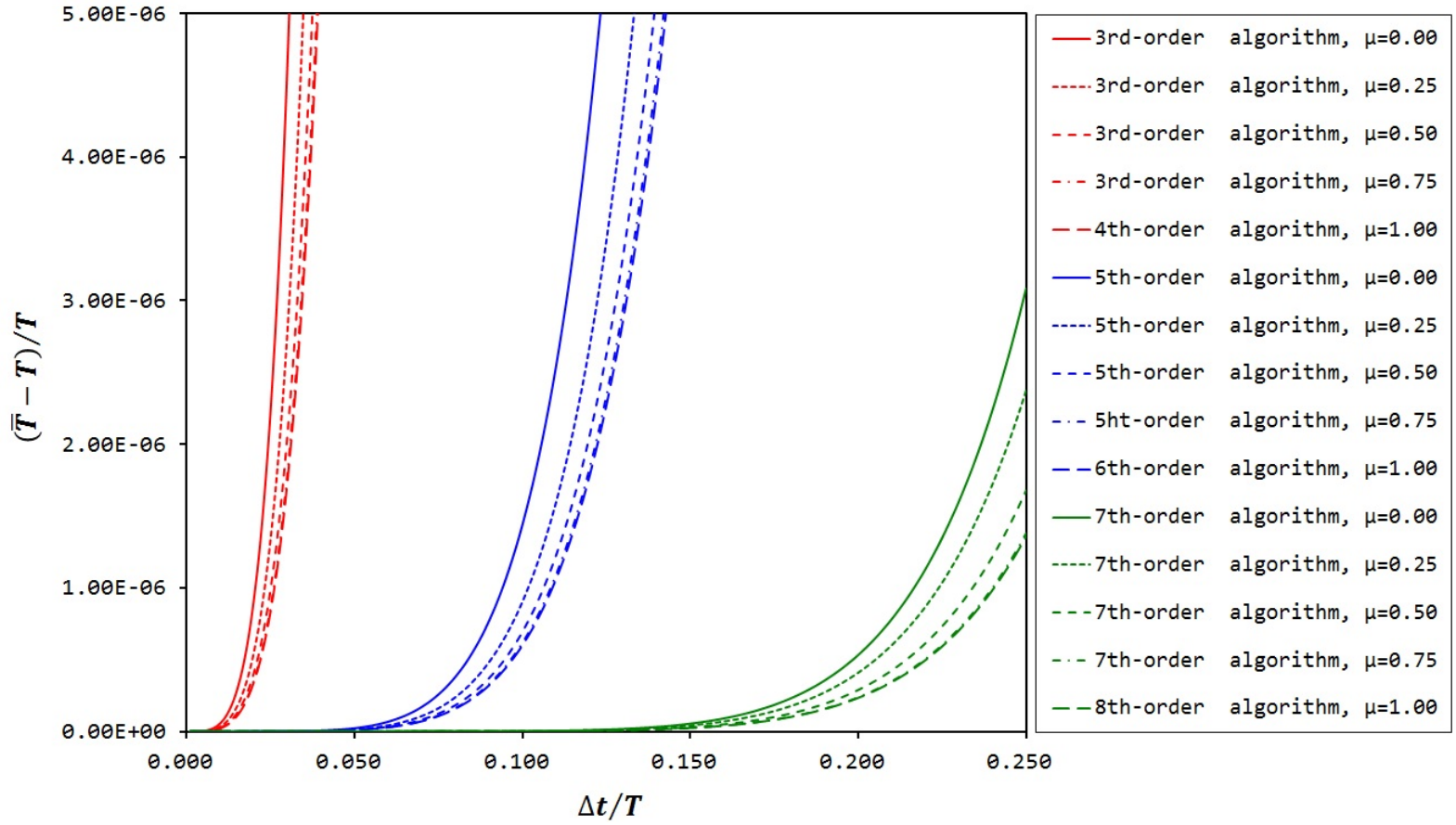

Figure 3.6: Relative period errors versus $\Delta t/T$ for current 3rd-, 5th-, and 7th-order algorithms with varying values of μ.

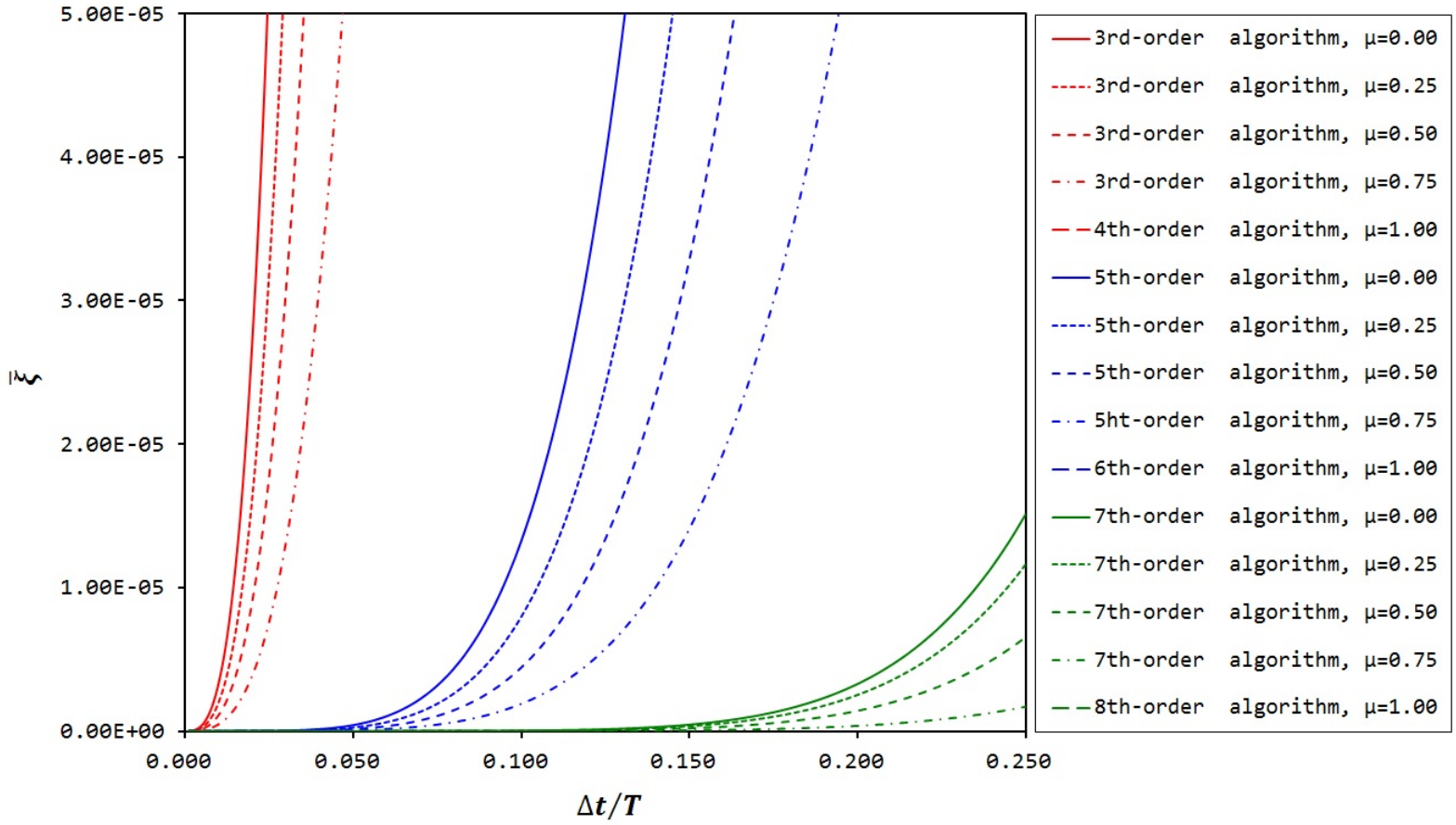

Figure 3.7: Damping ratios versus $\Delta t/T$ for current 3rd-, 5th-, and 7th-order algorithms with varying values of μ.

3.4 Numerical Examples

3.4.1 Multi-Degree-of-Freedom Spring

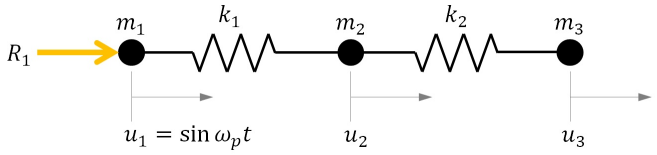

Figure 3.8: Description of three degrees of freedom spring system used by Bathe and Nho.

Bathe and Noh used the three degrees of freedom spring system shown in Fig. 3.8 to represent a simplified version of the complex structural system, which consists of stiff and flexible parts. The governing equation of the spring system shown in Fig. 3.8 is given by

$$\begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix} \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \end{Bmatrix} + \begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1 + k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} R_1 \\ 0 \\ 0 \end{Bmatrix} \tag{3.44}$$

where $u_1 = \sin \omega_p t$ and R_1 is the reaction force at node 1. The initial conditions of the zero displacement, velocity, and acceleration vectors (i.e., $u_i = 0$, $v_i = 0$ and $a_i = 0$ for $i = 1, 2, 3$) are used. Then by using the prescribed displacement at the first node, Eq. (3.44) can be reduced to

$$\begin{bmatrix} m_2 & 0 \\ 0 & m_3 \end{bmatrix} \begin{Bmatrix} a_2 \\ a_3 \end{Bmatrix} + \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix} \begin{Bmatrix} u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} k_1 u_1 \\ 0 \end{Bmatrix} \tag{3.45}$$

The high frequency filtering capability of the newly developed higher-order algorithms are tested with the multi-degree-of-freedom spring problem presented in Fig. 3.8. Here, we solve Eq. (3.45) by using the newly developed higher-order algorithms, the generalized-α method [17], the trapezoidal rule [23], and the Baig and Bathe method [35]. We use $k_1 = 10^6$, $k_2 = 1$, $m_1 = 0$, $m_2 = 1$, $m_3 = 1$, and $\omega_p = \pi$ as the data of the problem. With the zero initial conditions, the exact solution of the second node can be obtained by using the modal analysis as follows:

$$\begin{aligned} u_2 =& 1.0000290137708212165 \sin(3.1415926535897932385\, t) \\ & - 0.00000597923769734078593 95 \sin(0.99999949999987499995\, t) \\ & - 0.003141618947704035637 3 \sin(1000.0005000003749998\, t) \end{aligned} \tag{3.46}$$

To verify the high frequency filtering capability of the newly developed algorithms, numerically obtained solutions are compared with the reference solution which was obtained by eliminating the high frequency mode from the exact modal solution given in Eq. (3.46). As expected in the analysis of algorithms, Figs. 3.9-3.12 shows that the current higher-order algorithms can filter out the high frequency very effectively. Numerical displacement solutions obtained from various methods do not present noticeable differences, but the current algorithms can provide more accurate velocity and acceleration solutions than the existing second-order algorithms as presented in Figs. 3.10 and 3.12.

3.4.2 Two Dimensional Standing Wave Problem

The free vibration type of 2-D standing wave problem is analysed and numerical solutions are compared with each other to demonstrate the advantage of using the newly developed higher-order algorithms in long-term analysis. For this analysis, the 8th order algorithm (i.e., the 7th-order algorithm with $\mu = 1$) presented in Eq. (3.40)

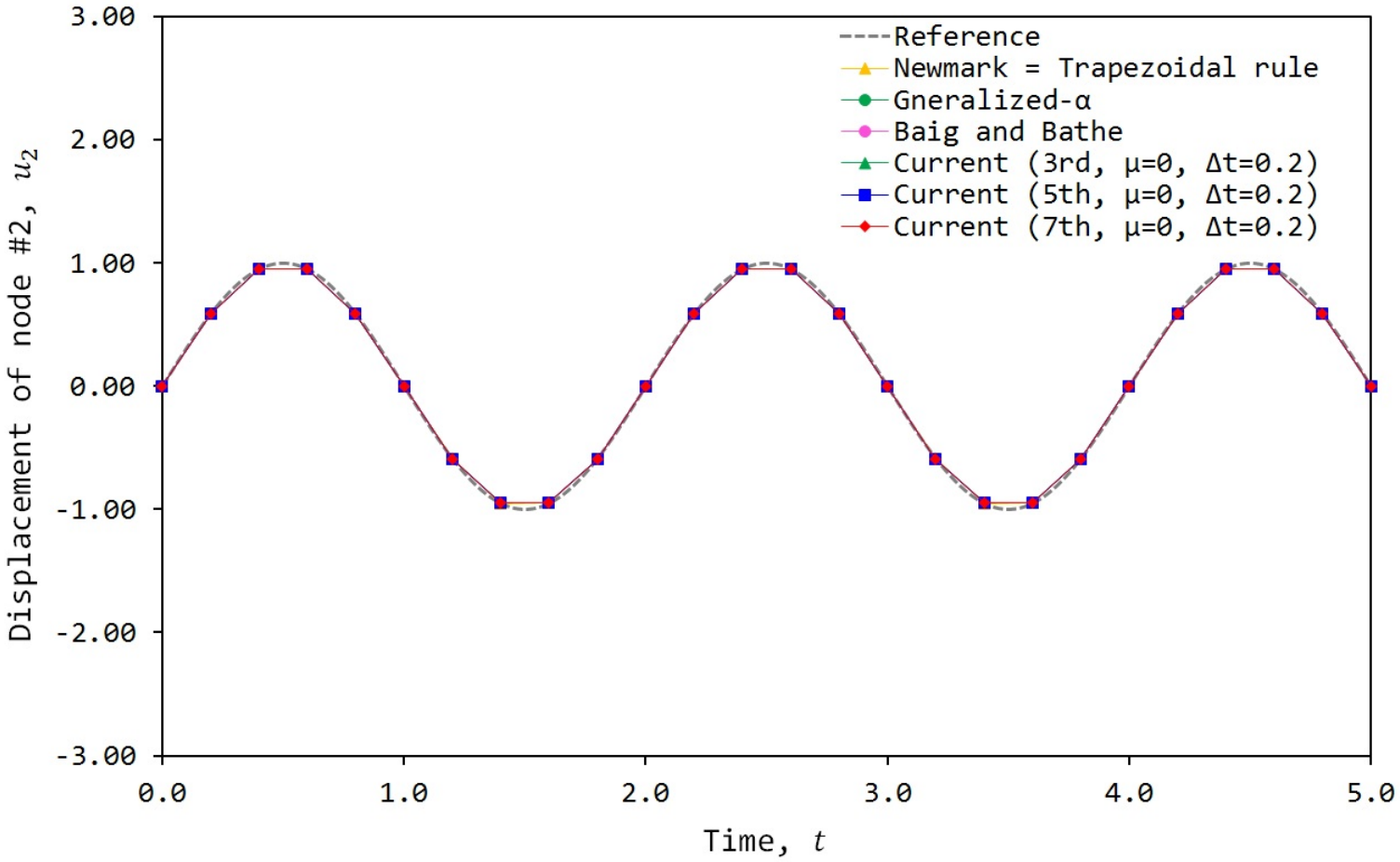

Figure 3.9: Displacement of second node for various methods.

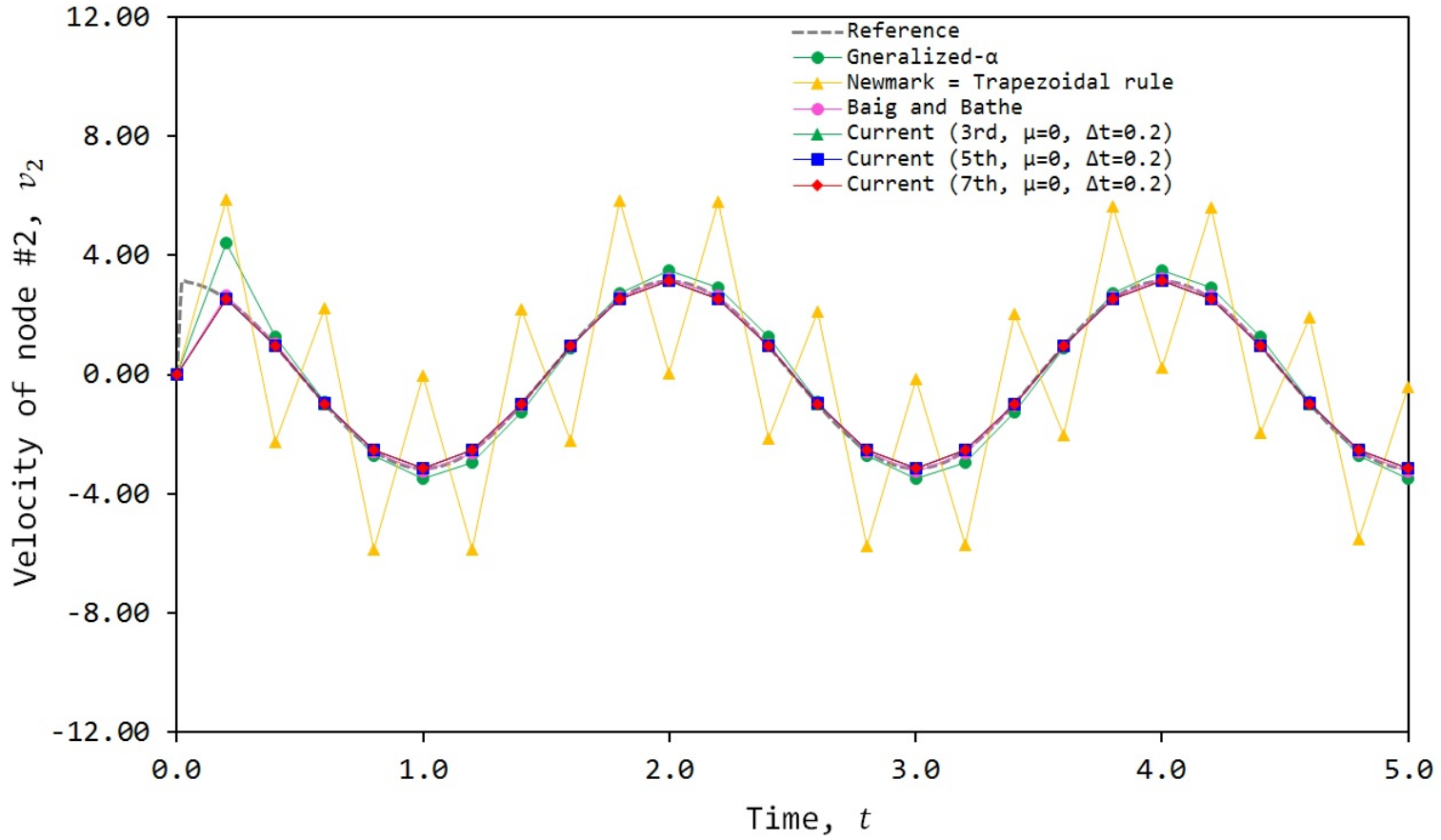

Figure 3.10: Velocity of second node for various methods.

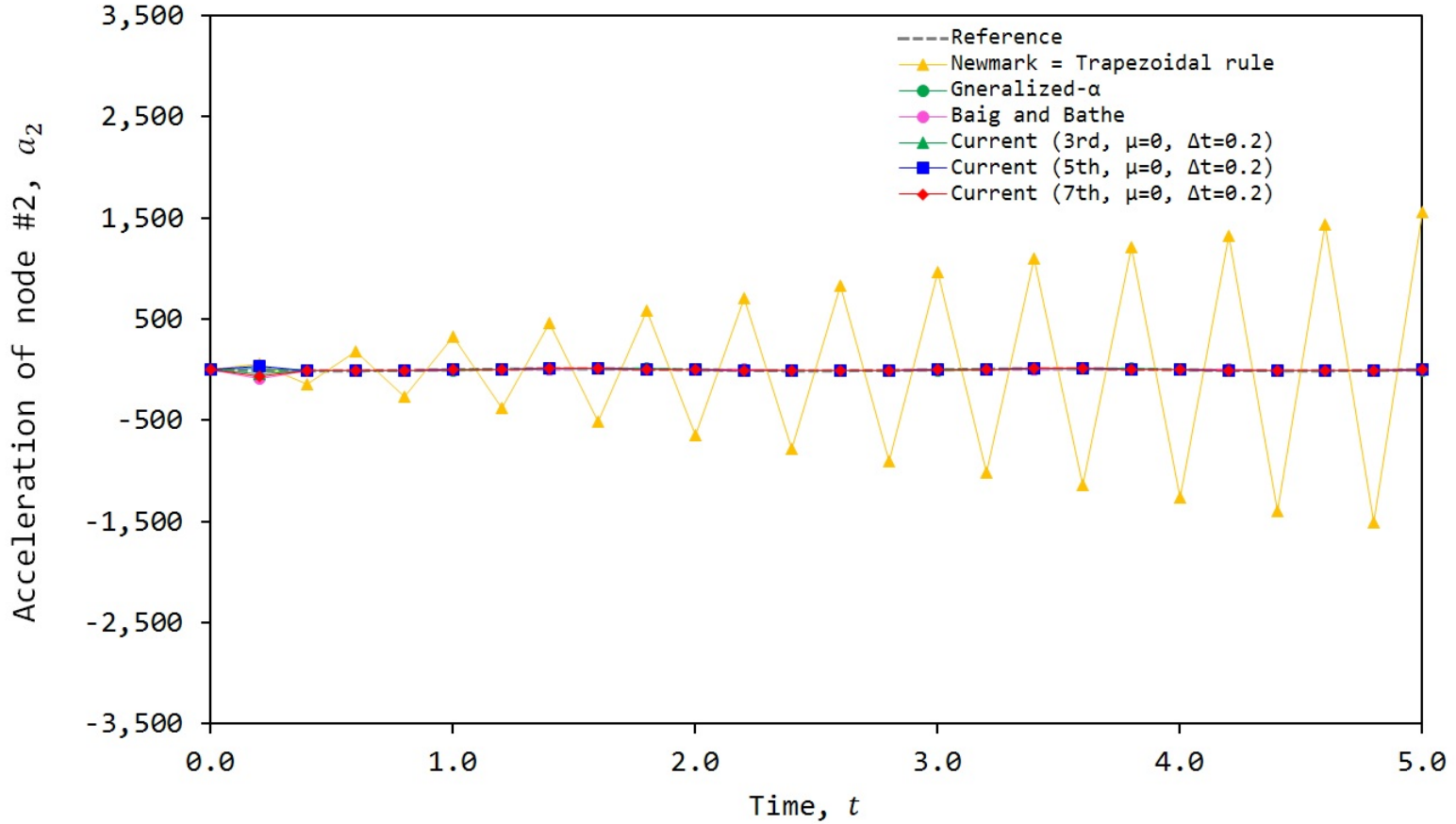

Figure 3.11: Acceleration of second node for various methods.

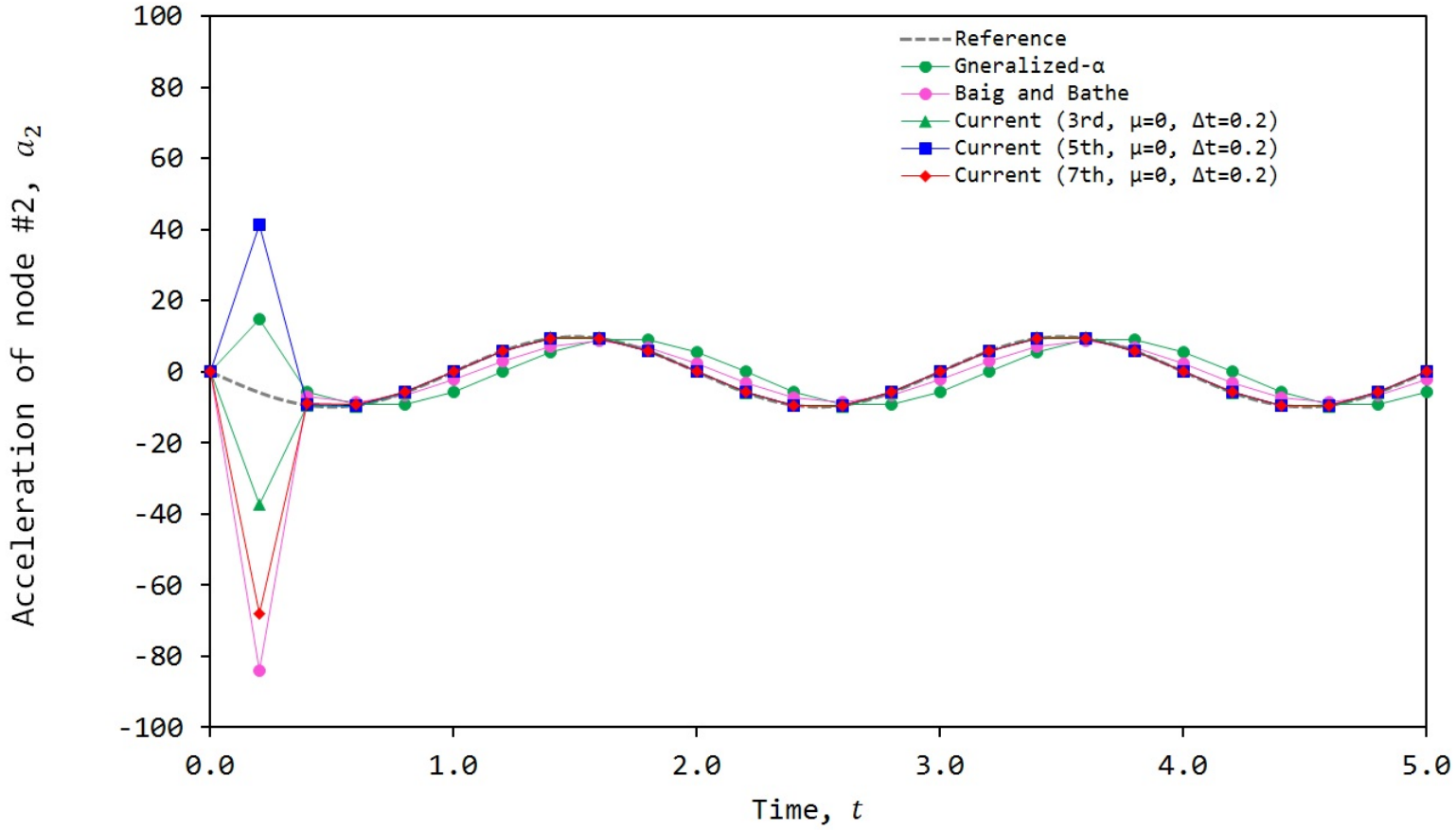

Figure 3.12: Acceleration of second node for various methods (enlarged image of Fig. 3.11 with out the trapezoidal rule).

is used.

The 2-D wave equation[82] is given by

$$\rho \frac{\partial^2 u}{\partial t^2} - \frac{\partial}{\partial x}\left(t_x \frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(t_y \frac{\partial u}{\partial y}\right) = 0 \tag{3.47}$$

with $t_x = t_y = \rho = 12.5$.

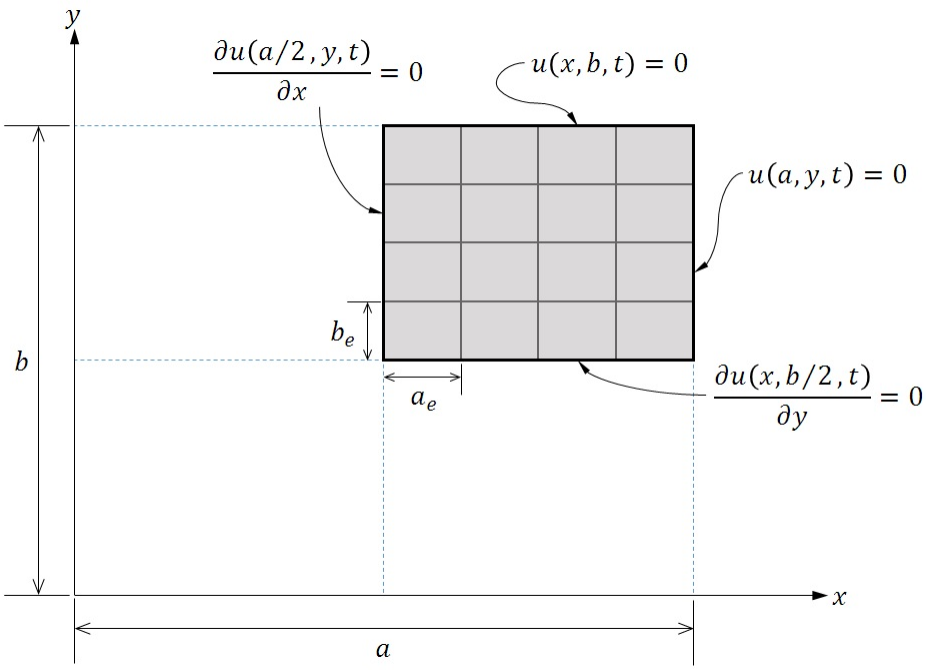

Figure 3.13: Computational domain and mesh used to analyse 2D wave equation.

For the spatial discretization, the weak form Galerkin method is used, and total 16 (4 by 4) quadratic (9-node) elements are used. The right top quadrant of the whole domain is chosen as the computational domain with the biaxial symmetric boundary conditions, as shown in Fig. 3.13. To investigate the effect of mass lumping, the original and lumped mass matrices are used in the analysis, and obtained numerical

solutions are compared. As presented in Fig. 3.13, the symmetry boundary conditions are chosen as

$$u(a,y,t) = u(x,b,t) = \frac{\partial u(a/2,y,t)}{\partial x} = \frac{\partial u(x,b/2,t)}{\partial y} = 0 \tag{3.48}$$

where $a = 2.0$ and $b = 1.0$. The initial conditions are taken as

$$u(x,y,0) = \frac{409.6}{\pi^6} \sum_{m,n=1,3,5\cdots}^{n} \left\{ \frac{1}{m^3 n^3} \sin\left(\frac{m\pi x}{4}\right) \sin\left(\frac{n\pi y}{2}\right) \right\} \tag{3.49}$$
$$\dot{u}(x,y,0) = 0$$

The series solution for this 2-D problem is given by

$$u(x,y,t) = \frac{409.6}{\pi^6} \sum_{m,n=1,3,5\cdots}^{n} \left\{ \frac{1}{m^3 n^3} \cos\left(\frac{\pi}{4}\sqrt{5(m^2+4n^2)}t\right) \sin\left(\frac{m\pi x}{4}\right) \sin\left(\frac{n\pi y}{2}\right) \right\} \tag{3.50}$$

In this numerical experiment, the 8th-order algorithm used $\Delta t = T/8$, whereas the trapezoidal rule used $\Delta t = T/80$, T being the period of the analytical solution. Numerical solutions are compared with the analytical solution given in Eq. (3.50) as presented in Figs. 3.14 and 3.15. As explained previously, each numerical solutions superposed the analytical solutions at the beginning of vibration as presented in Fig. 3.14. However, numerical solution obtained from the trapezoidal rule presented noticeable period error after hundred cycles of vibration as presented in Fig. 3.15.

The numerical solutions obtained from the case of the row proportional mass lumping [1, 96] have been also presented in Tables 3.1 and 3.2. In this particular example, use of the lumped mass matrix did not noticeably decrease the quality of numerical solutions. Thus, a proper mass lumping technique can be used in the

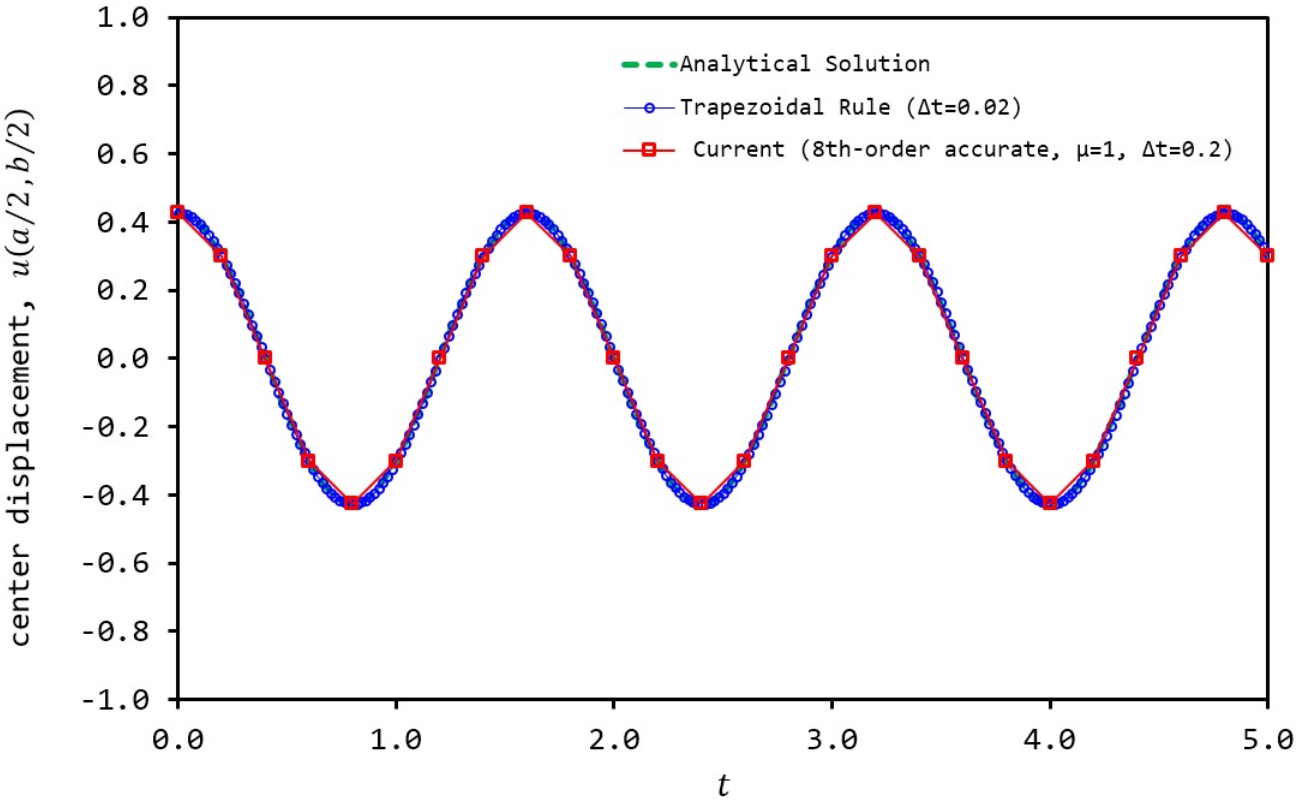

Figure 3.14: Comparison of center displacements at $0 \leq t \leq 5$ for various methods.

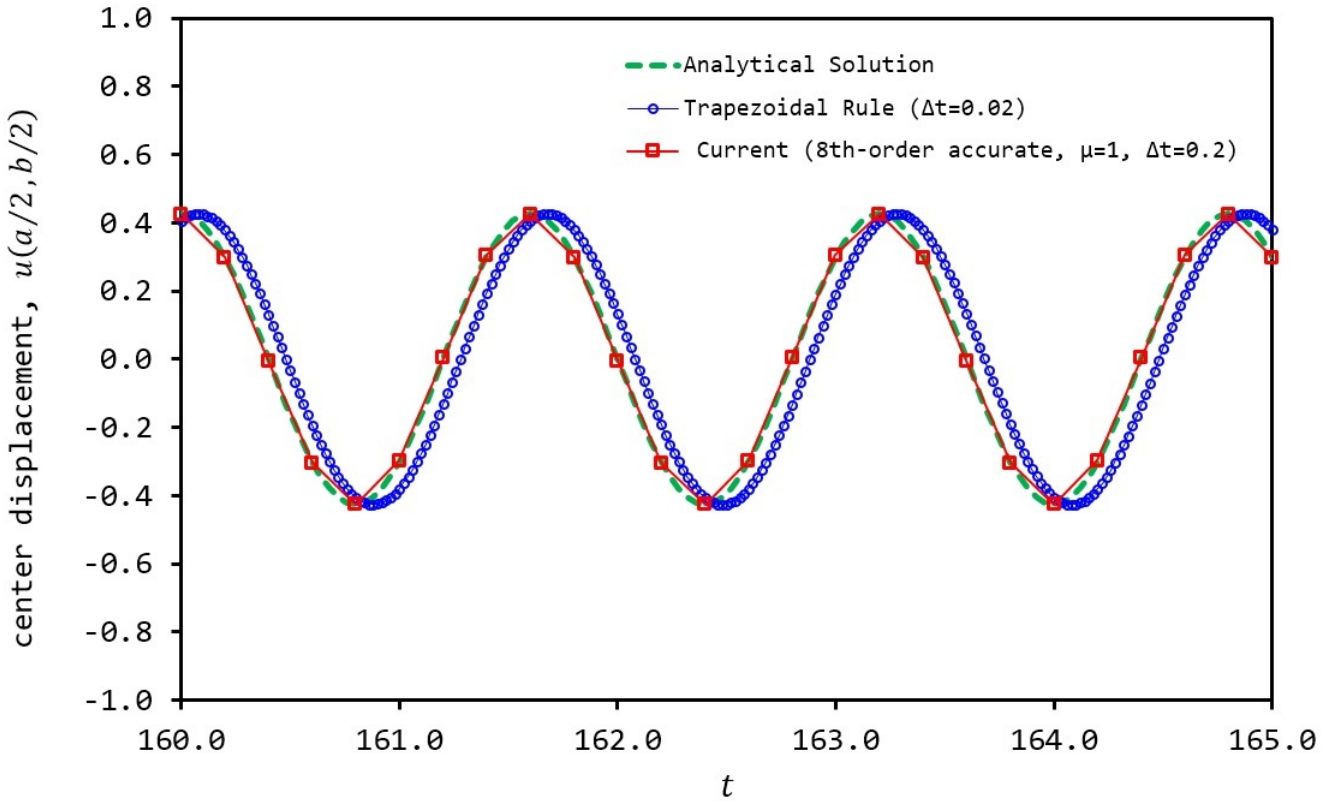

Figure 3.15: Comparison of center displacements at $160 \leq t \leq 165$ for various methods.

t	exact	with original mass matrix	with lumped mass matrix
0.2	0.30083333	0.30083316	0.30073439
0.4	0.00000000	-0.15477e-4	-0.10374e-3
0.6	-0.30083333	-0.30085062	-0.30063970
0.8	-0.42666667	-0.42667772	-0.42645930
1.0	-0.30083333	-0.30081412	-0.30091655
1.2	0.00000000	-0.24913e-4	-0.28491e-4
1.4	0.30083333	0.30087182	0.30076441
1.6	0.42666667	0.42667119	0.42642096

Table 3.1: Comparison of center displacements of 2D wave problems for current 8th-order algorithm with original and lumped mass matrices ($\Delta t = 0.2$).

t	exact	with original mass matrix	with lumped mass matrix
0.2	0.30083333	0.31209446	0.311900422
0.4	0.00000000	0.30929e-1	0.309691e-1
0.6	-0.30083333	-0.26574260	-0.26555092
0.8	-0.42666667	-0.42096253	-0.42094829
1.0	-0.30083333	-0.35196280	-0.35180152
1.2	0.00000000	-0.94094e-1	-0.941377e-1
1.4	0.30083333	0.21458218	0.21432998
1.6	0.42666667	0.40769570	0.40761813

Table 3.2: Comparison of center displacements of 2D wave problems for Newmark method (the trapezoidal rule) with original and lumped mass matrices ($\Delta t = 0.2$).

current algorithms if both efficiency and accuracy are required by the nature of the analysis.

3.5 Conclusion

In this study, the modified weighted residual method and the weight parameters have been used for the development of the new higher-order time integration algorithms. The displacement vector has been approximated as the linear combination of the pth-degree Hermite interpolation functions and the corresponding nodal values of the displacement vector and its time derivatives. The equation of structural dynamics has been used directly to define the residual vector in time. Then, two different order time derivatives of the residual vector have been manipulated in the modified weighted residual statements. Also, the equation of structural dynamics and its time derivatives were evaluated at the time nodes and they were used to eliminate the nodal accelerations and higher-order time derivatives of the displacement vector included in the Hermite approximation. The procedure proposed in this study can be used to construct $(p-1)$th- and pth-order algorithms, if the pth-degree Hermite approximation is used.

As a result of the proposed procedure, very unique forms of the result equations have been obtained. The computer implementation and the equation-solving procedure can also be helped by the unique forms of the result equations presented in Eqs. (3.33) - (3.37). In some unconventional spatial discretizations, the mass matrix can be automatically constructed in a diagonal form. In these cases, the new algorithms become extremely efficient compared to conventional higher-order algorithms. The numerical experiment conducted with the free vibration of the two-dimensional standing wave problem showed that the use of the lumped mass matrix did not altered accuracy of numerical solutions noticeably.

The result equations given in Eqs. (3.33) - (3.37) contained every information required for the computer implementation, which was not provided in the existing weighted residual method based algorithms. Thus, the new algorithms does not require any additional computation, such as reconstruction of solutions, to advance a step. As presented in Fig. 3.12, the new algorithms can filter out the spurious high frequency effects faster, at the same time, providing more accurate important low frequency solutions compared to the second-order algorithms. They also provided much better long-term solutions for the choice of large time steps as presented in Fig. 3.15.

In summary, the new algorithms can provide (a) unconditional stability, (b) controllable algorithmic dissipation, (c) easy computer implementation, and (d) efficient equation-solving. It is possible to apply the new algorithms to nonlinear analyses, if the governing equation is rearranged in a proper form.

4. TIME FINITE ELEMENT METHOD III

4.1 Introduction

Recently, various numerical methods have been used for the development of improved higher-order time integration algorithms which can be applied to the analysis of structural dynamics. These numerical methods include the Newmark approximation based methods [37, 38] , the weighted residual method [36, 60], the collocation method [63, 64], the differential quadrature method [66, 44], and the variational method [56, 57]. In general, these methods can be used to develop time integration algorithms of general $(2n-1)$th and $(2n)$th-order accuracy, n being the number of unknown vectors to be determined.

One common strategy of analysing partial differential equations (PDEs) associated with structural dynamics is to discretize the spatial domain of the PDEs first based on separation of variables [1]. After spatially discretizing the PDEs by employing proper numerical methods, a set of ordinary differential equations (ODEs) in time is obtained. This set of ODEs is called the semi-discrete equation of motion [1, 5] or the equation of linear structural dynamics [7, 8]. If the PDEs are linear, the semi-discrete system can be written as

$$\mathbf{M}\ddot{\mathbf{u}}(t) + \mathbf{C}\dot{\mathbf{u}}(t) + \mathbf{K}\mathbf{u}(t) = \mathbf{f}(t) \tag{4.1}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{C}$ is the viscous damping matrix, $\mathbf{K}$ is the stiffness matrices, $\mathbf{f}(t)$ is the vector of applied forces, $\mathbf{u}(t)$ is the displacement vector, $\dot{\mathbf{u}}(t)$ is the velocity vector, and $\ddot{\mathbf{u}}(t)$ is the acceleration vector. A solution of the initial value

problem described by Eq. (4.1) satisfies the following initial conditions

$$\mathbf{u}(0) = \mathbf{u}_0 \tag{4.2a}$$

$$\dot{\mathbf{v}}(0) = \mathbf{v}_0 \tag{4.2b}$$

where $\mathbf{u}_0$ and $\mathbf{v}_0$ are the initial displacement and velocity vectors, respectively.

According to Ref. [33], preferable attributes of higher-order time integration algorithms include unconditional stability, controllable algorithmic dissipation, $(2n-1)$th- and $(2n)$th-order accuracy, and full extensibility to nonlinear cases. Among many of the existing higher-order algorithms, only the algorithms developed based on the collocation method [71] and the modified differential quadrature method [69, 44] can satisfy all preferable attributes listed above. Especially, Fung's modified differential quadrature method can be used not only for the analysis structural dynamics (i.e., the second-order initial value problem), but also for other types of initial value problems, which is not possible in the collocation method. Thus, it can be said that the time integration algorithms developed based on Fung's modified differential quadrature method have the best extensibility among many of the existing higher-order algorithms as discussed in Ref. [44].

In Fung's modified quadrature method, $(2n-1)$th- and $(2n)$th-order accurate algorithms are obtained if n unknown displacement vectors and corresponding n sampling points are used in the developing procedure. However, optimized sampling points should be determined by finding n roots of the nth-degree polynomial equation whose coefficients contain a free parameter that controls the algorithmic dissipation. If the required order of accuracy is higher than fifth (i.e., $n \geq 5$), then the n roots of the nth-degree polynomial equation can be found only numerically for

every specification of algorithmic dissipation level. In addition to determining the sampling points, the last sampling point of Fung's modified differential quadrature method does not match the end point of the time interval (i.e., $t_n \neq t_{s+1}$). Due to the mismatch between the last sampling point and the end point of the time interval, n unknown displacement vectors should be found by solving a fully discrete system first, then they must be properly interpolated to compute the solution at the end of time interval to advance a step.

The purpose of this study is to develop a new family of higher-order time integration algorithms which can eliminate two additional procedures required in the higher-order algorithms developed based on Fung's modified quadrature method, while imitating all preferable features of them. In this study, we present a systematic and unified procedure which can be used for the development of general $(2n-1)$th-and $(2n)$th-order accurate algorithms, n being the number of unknown displacement vectors included in the time finite element approximation.

To this end, a modified weighted residual method which minimizes the residual vectors and time derivatives of them is considered. We note that Eq. (4.1) is not directly used in our residual minimization procedure. Instead of directly using Eq. (4.1) to define the residual vector in time, we rewrite Eq. (4.1) as a set of two first-order differential equations and one algebraic equation by introducing two additional variables (i.e., the velocity and acceleration vectors). In our study, the rewritten set of equations can be called the mixed formulations, and newly introduced velocity-displacement and acceleration-velocity relations in the mixed formulations are used to defined two residual vectors in time. Then the two residual vectors can be directly manipulated in the modified weighted residual statements to find discrete relations between included time dependent variables (i.e., the displacement, velocity and acceleration vectors).

In our case, the unique and advantageous computational structures of the algorithms are achieved through the unconventional manipulation of the velocity-displacement and acceleration-velocity relations in the mixed formulations. Also, the weight parameters are used to restate the integral form of the weighted residual statements as algebraic forms. Then, all preferable attributes of higher-order time integration algorithms are achieved through the optimization of the weight parameters. Through the optimization procedure, all weight parameters are stated in terms of one free parameter which can be used for the specification of the algorithmic dissipation levels. Since we use the equal nth-degree Lagrange interpolation functions for the approximations of all participating variables, determining sampling points and interpolating solutions are not necessary.

4.2 Development

New higher-order time integration algorithms can be systematically developed by employing the finite element method for a typical time domain. In the time finite element method, any time dependent variables can be approximated over the time interval as linear combinations of interpolation functions and nodal values of time dependent variables. Here, forms of approximations are very important, because specific choices of approximations can affect both computational structure and performance of time integration algorithms. The schematic concepts of time elements are presented in Figs. 4.1 and 4.2. As presented in Fig. 4.2, multiple choices are allowed for spacings of nodes in the Lagrange interpolation functions. In this study, the equally spaced nodes and the Gauss-Lobatto quadrature based nodes are considered for the higher-degree Lagrange interpolation functions. We note that the use of the Gauss-Lobatto quadrature points based nodes can improve the accuracy of particular solutions in the presence of higher-order excitations.

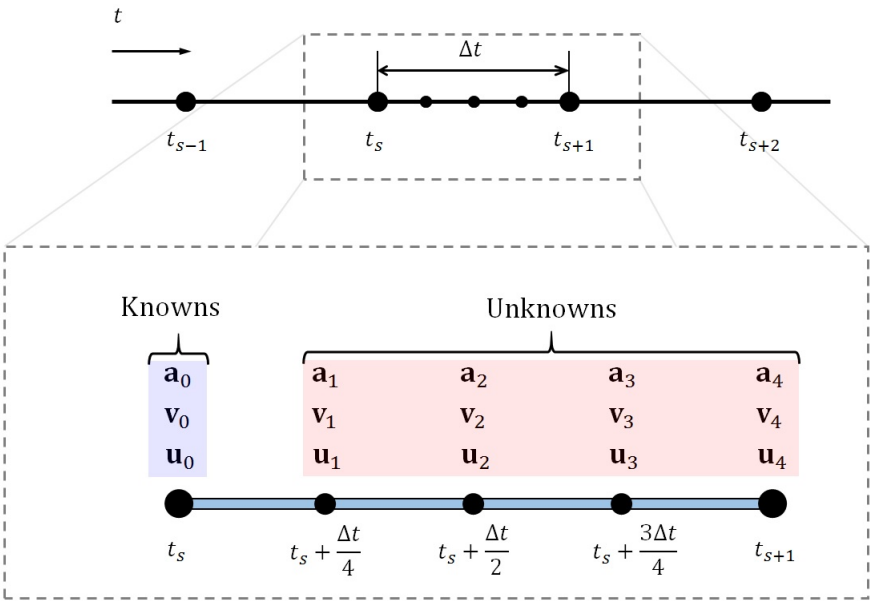

Figure 4.1: Schematic presentation of the time element obtained from the equally spaced 4th-degree Lagrange interpolations functions.

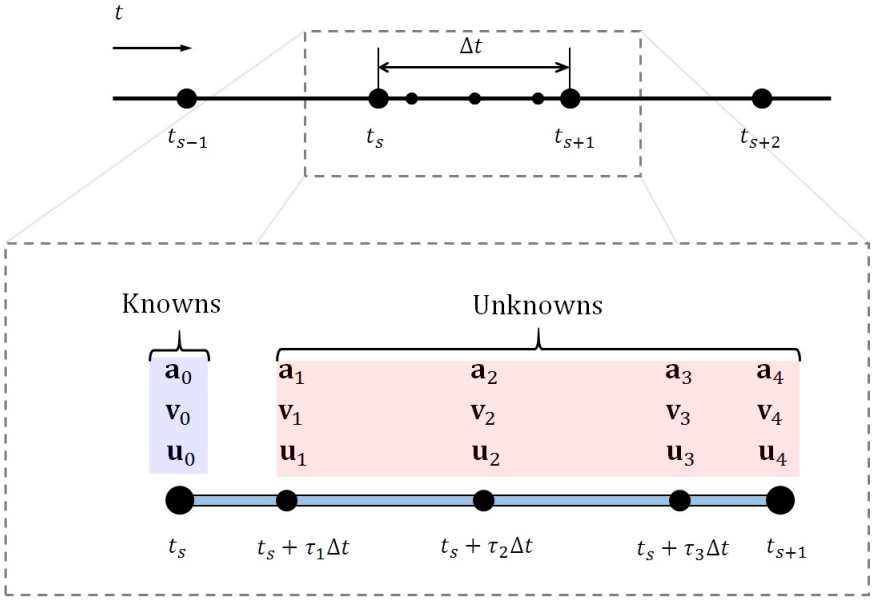

Figure 4.2: Schematic presentation of the time element obtained from the Gauss-Lobatto points based 4th-degree Lagrange interpolations functions.

4.2.1 Lagrange Approximations in Time

We can rewrite the equation of structural dynamics given in Eq. (4.1) as

$$\mathbf{M}\mathbf{a}(t) + \mathbf{C}\mathbf{v}(t) + \mathbf{K}\mathbf{u}(t) = \mathbf{f}(t) \tag{4.3a}$$

$$\mathbf{v}(t) = \dot{\mathbf{u}}(t) \tag{4.3b}$$

$$\mathbf{a}(t) = \dot{\mathbf{v}}(t) \tag{4.3c}$$

where, $\mathbf{v}(t)$ and $\mathbf{a}(t)$ are the newly introduced velocity and acceleration vectors in addition to the displacement vector $\mathbf{u}(t)$. A new family of higher-order algorithms can be developed based on Eqs. (4.3a)-(4.3c). Here, Eqs. (4.3a)-(4.3c) are called the mixed formulations because they contain three different types of time dependent variables (i.e., $\mathbf{u}(t)$, $\mathbf{v}(t)$ and $\mathbf{a}(t)$). It can be observed that the time differentiations of $\ddot{\mathbf{u}}(t)$ and $\dot{\mathbf{u}}(t)$ in Eq. (4.1) have been moved to the additional relations given in Eqs. (4.3b) and (4.3c). Thus, Eqs. (4.3b) and (4.3c) should be properly discretized in time, while Eq. (4.3a) can be used for the setting of dynamic equilibrium equations. Due to the use of the mixed formulations given in Eqs. (4.3a)-(4.3c), equal nth-degree Lagrange type interpolation functions can be used for the approximations of $\mathbf{u}(t)$, $\mathbf{v}(t)$ and $\mathbf{a}(t)$ to develop $(2n-1)$th- and $(2n)$th-order accurate algorithms. Over the time domain $t_s \leq t \leq t_{s+1}$, $\mathbf{u}(t)$, $\mathbf{v}(t)$, and $\mathbf{a}(t)$ can be independently approximated as

$$\mathbf{u}(t) \cong \bar{\mathbf{u}}(t) = \sum_{j=0}^{n} \psi_j(t)\mathbf{u}_j \tag{4.4a}$$

$$\mathbf{v}(t) \cong \bar{\mathbf{v}}(t) = \sum_{j=0}^{n} \psi_j(t)\mathbf{v}_j \tag{4.4b}$$

$$\mathbf{a}(t) \cong \bar{\mathbf{a}}(t) = \sum_{j=0}^{n} \psi_j(t)\mathbf{a}_j \tag{4.4c}$$

where $\psi_j(t)$ is the nth-degree Lagrange interpolation function associated with the jth node in the time element; $\mathbf{u}_j$, $\mathbf{v}_j$, and $\mathbf{a}_j$ are the displacement, velocity and acceleration vectors associated with the jth node in the time element, respectively.

In a general form, $\psi_j(t)$ can be written as

$$\psi_j(t) = \prod_{\substack{k=1 \\ (k\neq j)}}^{n} \frac{\bar{t} - \tau_k}{\tau_j - \tau_k} \tag{4.5}$$

where $\bar{t} = \frac{t-t_s}{\Delta t}$, Δt being the size of the time interval which is defined as $\Delta t = t_{s+1} - t_s$; τ_j is the parameter which determines the location of the jth time node as $t_j = t_s + \tau_j \, \Delta t$. In this study, τ_0 and τ_n are always 0 and 1, respectively.

4.2.2 Modified Weighted Residual Statement

As mentioned previously, a modified approach based on the weighted residual method is considered for the development of new algorithms. By substituting Eqs. (4.4a) - (4.4c) into Eqs. (4.3b) and (4.3c), two residual vectors (in time) can be defined as

$$\mathbf{r}_1(t) = \bar{\mathbf{v}}(t) - \dot{\bar{\mathbf{u}}}(t) \tag{4.6a}$$

$$\mathbf{r}_2(t) = \bar{\mathbf{a}}(t) - \dot{\bar{\mathbf{v}}}(t) \tag{4.6b}$$

Then, fully discretized relations can be found by minimizing $\mathbf{r}_1$ and $\mathbf{r}_2$ over the time domain ($t_s \leq t \leq t_{s+1}$) in the weighted integral sense. If the traditional weighted residual method is employed, n linearly independent weight functions are required

accordingly. The traditional weighted residual statements for $\mathbf{r}_1$ can be written as

$$\int_{t_s}^{t_{s+1}} w_i(t)\mathbf{r}_1(t)\,\mathrm{d}t = \mathbf{0} \quad \text{for} \quad i = 1, 2, ..., n \tag{4.7}$$

where n is the degree of the Lagrange interpolation functions, and $w_i(t)$ is the ith weight function in time. In the traditional weighted residual method, n linearly independent weight functions are required to state $\mathbf{v}_1, ..., \mathbf{v}_n$ in terms of $\mathbf{u}_1, ..., \mathbf{u}_n$ and the initial conditions ($\mathbf{u}_0$ and $\mathbf{v}_0$).

Our study, however, uses the modified weighted residual statements, which minimize the $(i-1)$th-order time derivative of the residual vectors given in Eqs. (4.6a) and (4.6b). With this unconventional approach, only one weight function can be used for each of the weighted residual residual statements given in Eqs. (4.6a) and (4.6b), since the linearly independencies are obtained through different orders of differentiations of the residual vectors. The modified weighted residual statements can be written as

$$\int_{t_s}^{t_{s+1}} w(t)\,\frac{\mathrm{d}^{i-1}}{\mathrm{d}t^{i-1}}\mathbf{r}_1(t)\,\mathrm{d}t = \mathbf{0} \quad \text{for} \quad i = 1, 2, ..., n \tag{4.8a}$$

$$\int_{t_s}^{t_{s+1}} w(t)\,\frac{\mathrm{d}^{i-1}}{\mathrm{d}t^{i-1}}\mathbf{r}_2(t)\,\mathrm{d}t = \mathbf{0} \quad \text{for} \quad i = 1, 2, ..., n \tag{4.8b}$$

where, n linearly independent relations are obtained through $\frac{\mathrm{d}^{i-1}}{\mathrm{d}t^{i-1}}\mathbf{r}_1(t)$ and $\frac{\mathrm{d}^{i-1}}{\mathrm{d}t^{i-1}}\mathbf{r}_2(t)$, respectively. The use of Eq. (4.8a) instead of Eq. (4.7) can reduce the number of weight parameters. As a consequence, the optimization procedure of these parameters can also be simplified.

4.2.3 Weight Parameters

In general, the weight function $w(t)$ is an arbitrary function in time, and it is not easy to assess the effect of changing $w(t)$ in Eqs. (4.8a) and (4.8b). However, the effects of $w(t)$ in Eqs. (4.8a) and (4.8b) can be assessed relatively easily if they are rewritten in algebraic forms by employing the weight parameters [59, 79]. By using the the local time $\tau = t - t_s$, the weight parameters can be defined as

$$\theta_k = \frac{\int_0^{\Delta t} w(\tau)\, \tau^k \, \mathrm{d}\tau}{\Delta t^k \int_0^{\Delta t} w(\tau)\, \mathrm{d}\tau} \quad \text{for} \quad k = 0, 1, 2, ..., n \tag{4.9}$$

Since $\theta_0 = 1$ for $k = 0$, we can restate the integral form of the modified weighted residual statements given in Eqs. (4.8a) and (4.8b) as algebraic forms by using n weight parameters (i.e., θ_1, θ_2, θ_3, ..., θ_n), while the traditional weighted residual statement given in Eq. (4.7) requires $n \times n$ weight parameters. Naturally, the optimization of the algorithms based on the traditional weighted residual method may become very complicated, because more weight parameters should be handled as the degree of the approximation increases.

To explain the developing procedure of the new algorithms in detail, we consider the case of the quadratic approximations (i.e., the case of $n = 2$). If $\mathbf{a}(t)$, $\mathbf{v}(t)$, and $\mathbf{u}(t)$ are approximated by using the quadratic Lagrange interpolation functions, Eq. (4.8a) can be restated in terms of the two weight parameters (i.e., θ_1 and θ_2) as follows:

$$\begin{aligned} &\int_0^{\Delta t} w(\tau)\, \mathbf{r}_1(\tau)\, \mathrm{d}\tau = \mathbf{0} \\ &\rightarrow (-2\,\theta_2 + 3\,\theta_1 - 1)\mathbf{v}_0 + (4\,\theta_2 - 4\,\theta_1)\mathbf{v}_1 + (-2\,\theta_2 + \theta_1)\mathbf{v}_2 \\ &\quad - \frac{1}{\Delta t}\left[(-4\,\theta_1 + 3)\mathbf{u}_0 + (8\theta_1 - 4)\mathbf{u}_1 + (-4\,\theta_1 + 1)\mathbf{u}_2\right] = \mathbf{0} \end{aligned} \tag{4.10a}$$

$$\int_0^{\Delta t} w(\tau)\,\frac{\mathrm{d}\mathbf{r}_1(\tau)}{\mathrm{d}t}\,\mathrm{d}\tau = \mathbf{0}$$
$$\rightarrow \frac{1}{\Delta t}\left[(-4\,\theta_1+3)\mathbf{v}_0 + (8\,\theta_1-4)\mathbf{v}_1 + (-4\,\theta_1+1)\mathbf{v}_2\right] \tag{4.10b}$$
$$-\frac{1}{\Delta t^2}\left(-4\,\mathbf{u}_0 + 8\,\mathbf{u}_1 - 4\,\mathbf{u}_2\right) = \mathbf{0}$$

Similarly, Eq. (4.8b) can also be written as

$$\int_0^{\Delta t} w(\tau)\,\mathbf{r}_2(\tau)\,\mathrm{d}\tau = \mathbf{0}$$
$$\rightarrow (-2\,\theta_2 + 3\,\theta_1 - 1)\mathbf{a}_0 + (4\,\theta_2 - 4\,\theta_1)\mathbf{a}_1 + (-2\,\theta_2 + \theta_1)\mathbf{a}_2 \tag{4.11a}$$
$$-\frac{1}{\Delta t}\left[(-4\,\theta_1+3)\mathbf{v}_0 + (8\theta_1-4)\mathbf{v}_1 + (-4\,\theta_1+1)\mathbf{v}_2\right] = \mathbf{0}$$

$$\int_0^{\Delta t} w(\tau)\,\frac{\mathrm{d}\mathbf{r}_2(\tau)}{\mathrm{d}t}\,\mathrm{d}\tau = \mathbf{0}$$
$$\rightarrow \frac{1}{\Delta t}\left[(-4\,\theta_1+3)\mathbf{a}_0 + (8\,\theta_1-4)\mathbf{a}_1 + (-4\,\theta_1+1)\mathbf{a}_2\right] \tag{4.11b}$$
$$-\frac{1}{\Delta t^2}\left(-4\,\mathbf{v}_0 + 8\,\mathbf{v}_1 - 4\,\mathbf{v}_2\right) = \mathbf{0}$$

In Eqs. (4.10) and (4.11), the vectors with subscript 0,1 and 2 denote that they are properties associated with time nodes t_s, $t_s + \tau_1\Delta t$ and $t_s + \tau_2\Delta t$, respectively. The discrete relations given in Eqs. (4.10) and (4.11) can be rearranged in more convenient forms of

$$\begin{Bmatrix}\mathbf{v}_1\\ \mathbf{v}_2\end{Bmatrix} = \frac{1}{\Delta t}\begin{bmatrix} -\frac{8\,\theta_1^2-4\,\theta_1-4\,\theta_2+1}{2\,\theta_1^2-\theta_2}\mathbf{I} & \frac{16\,\theta_1^2-4\,\theta_1-8\,\theta_2+1}{8\,\theta_1^2-2\,\theta_2}\mathbf{I} \\ -\frac{4(4\,\theta_1^2-2\,\theta_1-2\,\theta_2+1)}{2\,\theta_1^2-\theta_2}\mathbf{I} & \frac{8\,\theta_1^2-2\,\theta_1-4\,\theta_2+1}{2\,\theta_1^2-\theta_2}\mathbf{I}\end{bmatrix}\begin{Bmatrix}\mathbf{u}_1\\ \mathbf{u}_2\end{Bmatrix}$$
$$+\frac{1}{\Delta t}\begin{Bmatrix}\frac{16\,\theta_1^2-12\,\theta_1-8\,\theta_2+3}{8\,\theta_1^2-4\,\theta_2}\mathbf{I}\\ \frac{8\,\theta_1^2-6\,\theta_1-4\,\theta_2+3}{2\,\theta_1^2-\theta_2}\mathbf{I}\end{Bmatrix}\mathbf{u}_0 + \begin{Bmatrix}\frac{8\,\theta_1^2-4\,\theta_1-4\,\theta_2+1}{8\,\theta_1^2-4\,\theta_2}\mathbf{I}\\ \frac{2\,\theta_1^2-2\,\theta_1-\theta_2+1}{2\,\theta_1^2-\theta_2}\mathbf{I}\end{Bmatrix}\mathbf{v}_0 \tag{4.12a}$$

$$\begin{Bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \end{Bmatrix} = \frac{1}{\Delta t} \begin{bmatrix} -\frac{8\,\theta_1^2-4\,\theta_1-4\,\theta_2+1}{2\,\theta_1^2-\theta_2}\mathbf{I} & \frac{16\,\theta_1^2-4\,\theta_1-8\,\theta_2+1}{8\,\theta_1^2-2\,\theta_2}\mathbf{I} \\ -\frac{4(4\,\theta_1^2-2\,\theta_1-2\,\theta_2+1)}{2\,\theta_1^2-\theta_2}\mathbf{I} & \frac{8\,\theta_1^2-2\,\theta_1-4\,\theta_2+1}{2\,\theta_1^2-\theta_2}\mathbf{I} \end{bmatrix} \begin{Bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{Bmatrix} + \frac{1}{\Delta t} \begin{Bmatrix} \frac{16\,\theta_1^2-12\,\theta_1-8\,\theta_2+3}{8\,\theta_1^2-4\,\theta_2}\mathbf{I} \\ \frac{8\,\theta_1^2-6\,\theta_1-4\,\theta_2+3}{2\,\theta_1^2-\theta_2}\mathbf{I} \end{Bmatrix} \mathbf{v}_0 + \begin{Bmatrix} \frac{8\,\theta_1^2-4\,\theta_1-4\,\theta_2+1}{8\,\theta_1^2-4\,\theta_2}\mathbf{I} \\ \frac{2\,\theta_1^2-2\,\theta_1-\,\theta_2+1}{2\,\theta_1^2-\theta_2}\mathbf{I} \end{Bmatrix} \mathbf{a}_0 \tag{4.12b}$$

where $\mathbf{I}$ is an m by m identity matrix, m being the size of the equation of structural dynamics given in Eq. (4.1). To obtain the fully discrete equation of Eq. (4.1), two dynamic equilibrium equations can be obtained by evaluating Eq. (4.3a) at $t = t_s + \tau_1\ \Delta t$ and $t = t_s + \tau_2 \Delta t$, respectively. Then, those equilibrium equations can be rearranged as

$$\begin{bmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{0} & \mathbf{M} \end{bmatrix} \begin{Bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \end{Bmatrix} + \begin{bmatrix} \mathbf{C} & \mathbf{0} \\ \mathbf{0} & \mathbf{C} \end{bmatrix} \begin{Bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{Bmatrix} + \begin{bmatrix} \mathbf{K} & \mathbf{0} \\ \mathbf{0} & \mathbf{K} \end{bmatrix} \begin{Bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{Bmatrix} = \begin{Bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{Bmatrix} \tag{4.13}$$

where $\mathbf{f}_1 = \mathbf{f}(t_s + \Delta t/2)$ and $\mathbf{f}_2 = \mathbf{f}(t_s + \Delta t)$. By using Eqs. (4.12a) and (4.12b), $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{a}_1$, and $\mathbf{a}_2$ can be eliminated from Eq. (4.13). Then Eq. (4.13) can be solved to find $\mathbf{u}_1$ and $\mathbf{u}_2$. If the weight parameters (θ_1 and θ_2) are used without optimization, only second-order accuracy can be obtained and unconditional stability may not be guaranteed.

4.2.4 Optimization

Here, we present the optimization procedure of the weight parameters. We can optimize the weight parameters included in Eqs. (4.12a) and (4.12b) by using the single-degree-of-freedom problem of

$$\ddot{u}(t) + 2\,\xi\,\omega\,\dot{u}(t) + \omega^2\,u(t) = f(t) \tag{4.14}$$

with the initial conditions

$$u(0) = u_0 \tag{4.15a}$$

$$\dot{u}(0) = v_0 \tag{4.15b}$$

where, u_0 and v_0 are the initial displacement and velocity. For the homogeneous case (i.e., $f(t) = 0$), the exact solution of Eqs. (4.14)-(4.15) is given by

$$\begin{aligned} u\left(t\right) = &\exp^{(-\xi\omega t)}\left(\cos\left(\omega_d t\right) + \frac{\xi\omega}{\omega_d}\sin\left(\omega_d t\right)\right)u_0 \\ &+ \frac{1}{\omega_d}\exp^{(-\xi\omega t)}\sin\left(\omega_d t\right)v_0 \end{aligned} \tag{4.16}$$

and the first-order differentiation of $u\left(t\right)$ with respect to t gives the exact velocity of

$$\begin{aligned} \dot{u}\left(t\right) = &-\frac{\left(\xi^2\omega^2 + {\omega_d}^2\right)}{\omega_d}\exp^{(-\xi\omega t)}\sin\left(\omega_d t\right)u_0 \\ &+ \frac{1}{\omega_d}\exp^{(-\xi\omega t)}\Big(-\xi\omega\sin\left(\omega_d t\right) + \omega_d\cos\left(\omega_d t\right)\Big)v_0 \end{aligned} \tag{4.17}$$

where $\omega_d = \sqrt{1-\xi^2}\,\omega$. Now, the exact solutions given in Eqs. (4.16) and (4.17) can be used to write the exact discrete solution at $t = \Delta t$ as

$${}^{e}\mathbf{x}_1 = {}^{e}\mathbf{A}\mathbf{x}_0 \tag{4.18}$$

where ${}^{e}\mathbf{A}$, ${}^{e}\mathbf{x}_1$ and $\mathbf{x}_0$ are the exact amplification matrix, the exact solution vector and the initial condition vector. Here ${}^{e}\mathbf{A}$, ${}^{e}\mathbf{x}_1$ and $\mathbf{x}_0$ can be defined as

$${}^{e}\mathbf{x}_1 = \begin{Bmatrix} {}^{e}u(\Delta t) \\ {}^{e}\dot{u}(\Delta t) \end{Bmatrix}, \quad {}^{e}\mathbf{A} = \begin{bmatrix} {}^{e}A_{11}(\xi,\omega,\Delta t) & {}^{e}A_{12}(\xi,\omega,\Delta t) \\ {}^{e}A_{21}(\xi,\omega,\Delta t) & {}^{e}A_{22}(\xi,\omega,\Delta t) \end{bmatrix}, \quad \mathbf{x}_0 = \begin{Bmatrix} u_0 \\ v_0 \end{Bmatrix} \tag{4.19}$$

where the specific terms of ${}^{e}\mathbf{A}$ are

$$\begin{aligned}
{}^{e}A_{11} &= \exp^{(-\xi\omega\Delta t)}\left(\cos\left(\omega_d\Delta t\right) + \frac{\xi\omega}{\omega_d}\sin\left(\omega_d\Delta t\right)\right) \\
{}^{e}A_{12} &= \frac{1}{\omega_d}\exp^{(-\xi\omega\Delta t)}\sin\left(\omega_d\Delta t\right) \\
{}^{e}A_{21} &= -\frac{\left(\xi^2\omega^2 + {\omega_d}^2\right)}{\omega_d}\exp^{(-\xi\omega\Delta t)}\sin\left(\omega_d\Delta t\right) \\
{}^{e}A_{22} &= \frac{1}{\omega_d}\exp^{(-\xi\omega\Delta t)}\Big(-\xi\omega\sin\left(\omega_d\Delta t\right) + \omega_d\cos\left(\omega_d\Delta t\right)\Big)
\end{aligned} \tag{4.20}$$

Here ${}^{e}u(\Delta t)$ and ${}^{e}\dot{u}(\Delta t)$ are the exact displacement and velocity solutions at $t = \Delta t$.

Similarly, the discrete displacement and velocity solutions of Eq. (4.14) can be numerically obtained by using Eqs. (4.12b), (4.12b), and (4.13). Then, these numerically obtained discrete solutions can be rearranged as

$${}^{a}\mathbf{x}_1 = {}^{a}\mathbf{A}\mathbf{x}_0 \tag{4.21}$$

where ${}^{a}\mathbf{A}$ and ${}^{a}\mathbf{x}_1$ are the numerical amplification matrix and the discrete numerical solution vector, respectively, and $\mathbf{x}_0$ is the initial condition vector. For the current case of $n = 2$ (i.e., the quadratic approximations of dependent variables), ${}^{a}\mathbf{A}$ and ${}^{a}\mathbf{x}_1$ are defined as

$$\begin{gathered}
{}^{a}\mathbf{A} = \begin{bmatrix} {}^{a}A_{11}(\xi,\omega,\Delta t,\theta_1,\theta_2) & {}^{a}A_{12}(\xi,\omega,\Delta t,\theta_1,\theta_2) \\ {}^{a}A_{21}(\xi,\omega,\Delta t,\theta_1,\theta_2) & {}^{a}A_{22}(\xi,\omega,\Delta t,\theta_1,\theta_2) \end{bmatrix} \\
{}^{a}\mathbf{x}_1 = \begin{Bmatrix} {}^{a}u(\Delta t) \\ {}^{a}v(\Delta t) \end{Bmatrix}, \quad {}^{a}\mathbf{x}_0 = \begin{Bmatrix} u_0 \\ v_0 \end{Bmatrix}
\end{gathered} \tag{4.22}$$

where ${}^{a}u(\Delta t)$ and ${}^{a}v(\Delta t)$ are the numerical displacement and velocity solutions at $t = \Delta t$. For the current case, Eqs. (4.12)-(4.13) can be directly used to find ${}^{a}u(\Delta t)$

and ${}^{a}v(\Delta t)$. By setting $\mathbf{M} = 1$, $\mathbf{C} = 2\,\xi\,\omega$, $\mathbf{K} = \omega^2$, $\mathbf{u}_0 = u_0$, $\mathbf{v}_0 = v_0$, $\mathbf{a}_0 = -(2\,\xi\,\omega\,v_0 + \xi^2\,u_0)$, $\mathbf{u}_2 = {}^{a}u(\Delta t/2)$, $\mathbf{v}_1 = {}^{a}v(\Delta t/2)$, $\mathbf{u}_2 = {}^{a}u(\Delta t)$ and $\mathbf{v}_2 = {}^{a}v(\Delta t)$, we can directly obtain Eq. (4.21). For the numerical discrete solution given in Eq. (4.21) to be $(2n-1)$th-order accurate, the highest order term of Δt in the Taylor expansion of entries of ${}^{e}\mathbf{A} - {}^{a}\mathbf{A}$ should satisfy

$$ {}^{e}A_{ij} - {}^{a}A_{ij} = \mathrm{O}\left(\Delta t^{\,p+1}\right) \quad \text{for} \quad i,j = 1,2 \tag{4.23} $$

The numerical discrete solution given in Eq. (4.21) becomes only 2nd-order accurate without proper optimization of θ_1 and θ_2. The Taylor's expansion of each entry of ${}^{e}\mathbf{A} - {}^{a}\mathbf{A}$ can be computed as

$$ \begin{aligned} {}^{e}A_{11} - {}^{a}A_{11} &= \frac{1}{3}\omega^3\xi\left(6\theta_1^2 - 3\theta_1 - 3\theta_2 + 1\right)\Delta t^3 + \mathrm{O}\left(\Delta t^4\right) \\ {}^{e}A_{12} - {}^{a}A_{12} &= \frac{1}{6}\omega^2(2\xi-1)(2\xi+1)\left(6\theta_1^2 - 3\theta_1 - 3\theta_2 + 1\right)\Delta t^3 + \mathrm{O}\left(\Delta t^4\right) \\ {}^{e}A_{21} - {}^{a}A_{21} &= -\frac{1}{6}\omega^4(2\xi-1)(2\xi+1)\left(6\theta_1^2 - 3\theta_1 - 3\theta_2 + 1\right)\Delta t^3 + \mathrm{O}\left(\Delta t^4\right) \\ {}^{e}A_{22} - {}^{a}A_{22} &= -\frac{2}{3}\omega^3\xi(2\xi^2-1)\left(6\theta_1^2 - 3\theta_1 - 3\theta_2 + 1\right)\Delta t^3 + \mathrm{O}\left(\Delta t^4\right) \end{aligned} \tag{4.24} $$

To improve the order of accuracy from 2nd-order to 3rd-order, all terms with Δt^3 in Eq. (4.24) should be vanished. For this end, θ_2 can be chosen as

$$ \theta_2 = 2\,\theta_1^2 - \theta_1 + \frac{1}{3} \tag{4.25} $$

If the optimized θ_2 given in Eq. (4.25) is used, the Taylor expansion of ${}^{e}A_{ij} - {}^{a}A_{ij}$

becomes

$$\begin{aligned}
{}^{e}A_{11} - {}^{a}A_{11} &= -\frac{1}{24}\omega^4(4\xi^2-1)\,(2\theta_2-1)\,\Delta t^4 + \mathrm{O}\left(\Delta t^5\right) \\
{}^{e}A_{12} - {}^{a}A_{12} &= -\frac{1}{6}\omega^3\xi(2\xi^2-1)\,(2\theta_2-1)\,\Delta t^4 + \mathrm{O}\left(\Delta t^5\right) \\
{}^{e}A_{21} - {}^{a}A_{21} &= \frac{1}{6}\omega^5\xi(2\xi^2-1)\,(2\theta_2-1)\,\Delta t^4 + \mathrm{O}\left(\Delta t^5\right) \\
{}^{e}A_{22} - {}^{a}A_{22} &= \frac{1}{24}\omega^4(4\xi^2-2\xi-1)(4\xi^2+2\xi-1)\,(2\theta_2-1)\,\Delta t^4 + \mathrm{O}\left(\Delta t^5\right)
\end{aligned} \tag{4.26}$$

Here, θ_1 can be chosen as $\frac{1}{2}$, then we can eliminate terms with Δt^4 in Eq. (4.26), making the algorithm 4th-order accurate. However, this case will make the algorithm non-dissipative one, and we cannot control dissipations. Since we wish to have a control of algorithmic dissipation, we stop improving the accuracy of the algorithm, but θ_1 can be selected judiciously for the algorithmic dissipation control. The choice of θ_1 can be helped by manipulating the spectral radius in the high frequency limit. The spectral radius of ${}^{a}\mathbf{A}$ is defined as

$$\rho({}^{a}\mathbf{A}) = \max\left(|\lambda_1|, |\lambda_2|\right) \tag{4.27}$$

where λ_1 and λ_2 are two eigenvalues of ${}^{a}\mathbf{A}$. We note that $\rho({}^{a}\mathbf{A})$ is the measure of stability and algorithmic dissipation for varying $\frac{\Delta t}{T}$, T being the period of the given problem. By using $\rho({}^{a}\mathbf{A})$, the ultimate spectral radius is defined as

$$\mu = \lim_{\Omega\to\infty} \rho({}^{a}\mathbf{A}) \tag{4.28}$$

For a practical algorithmic dissipation control, we relate the last remaining weight parameter (θ_1) to the ultimate spectral radius (μ). By using the definition of the

ultimate spectral radius given in Eq. (4.28), μ can be computed as

$$\mu = \frac{-3\,\theta_1 + 2}{-1 + 3\,\theta_1} \tag{4.29}$$

From Eq. (4.29), θ_1 can be selected as

$$\theta_1 = \frac{(\mu + 2)}{3\,(\mu + 1)} \tag{4.30}$$

Since the unconditional stability condition requires the spectral radius to satisfy $0 \leq \rho({}^{a}\mathbf{A}) \leq 1$ for $\Delta t \geq$, μ can also be chosen from the range of $0 \leq \mu \leq 1$. It should also be noted that the choice of $\mu = 1$ gives $\theta_1 = 1/2$ for the case of $n = 2$, and $\mu = 1$ satisfies the 4th-order accuracy condition presented in Eq. (4.26).

In this section, we presented the optimization procedure of the quadratic approximation case as a simple example, but this procedure can be extended to general nth-degree approximations to get $(2n-1)$th order accurate algorithms. For nth-degree Lagrange type approximation, $(2n-1)$th-order accuracy can be improved up to $(2n)$th-order, if $\mu = 1$ is used.

4.2.5 Gauss-Lobbato Quadrature Points Based Lagrange Interpolation Functions

In the presence of higher-order excitations, such as the trigonometric and exponential functions, the algorithms obtained from the equally spaced nodal points cannot provide $(2n-1)$th- or $(2n)$th-order accuracy. However, it should be emphasized that this drawback of the algorithms based on the equally spaced Lagrange interpolation functions can be easily overcame by using the Gauss-Lobbato quadrature points based Lagrange interpolation functions. The algorithms obtained from the Gauss-Lobbato quadrature points based Lagrange interpolation functions [97] can provide $(2n-1)$th- and $(2n)$th-order accuracies for particular solutions in the

	Equally Spaced nodes	Gauss-Lobbato quadrature point based nodes
τ_0	0	0
τ_1	$\frac{1}{3}$	$-\frac{\sqrt{5}}{10}+\frac{1}{2}$
τ_2	$\frac{2}{3}$	$\frac{\sqrt{5}}{10}+\frac{1}{2}$
τ_3	1	1

Table 4.1: Comparison of nodal spacing parameter τ_i for equally spaced and Gauss-Lobbato quadrature points based 3rd-degree Lagrange interpolation functions

presence of higher-order excitations. In the nth degree Lagrange interpolation functions, the location of ith node can be expressed as

$$t_i = t_s + \tau_i \, \Delta t \quad \text{for} \quad i = 0, 1, 2, ..., n \tag{4.31}$$

where, τ_i is the ith node spacing parameter which specifies the location of ith time node in the time element, and n is the degree of the interpolation functions used. Specific values of the node spacing parameters for $n = 3, 4$ are presented in Tables 4.1 and 4.2 and the plots of the associated Lagrange interpolation functions are also presented in Figures 4.3-4.5.

It should also be noted that the optimization procedure of the weight parameters is not affected by the choice of the interpolation points in the proposed procedure. Thus, the optimized weight parameters obtained from the equally spaced Lagrange approximations case can also be used for the Gauss-Lobbato quadrature points based Lagrange approximations case, if the degree of approximations are the same. Some result equations of both cases are presented in the next section.

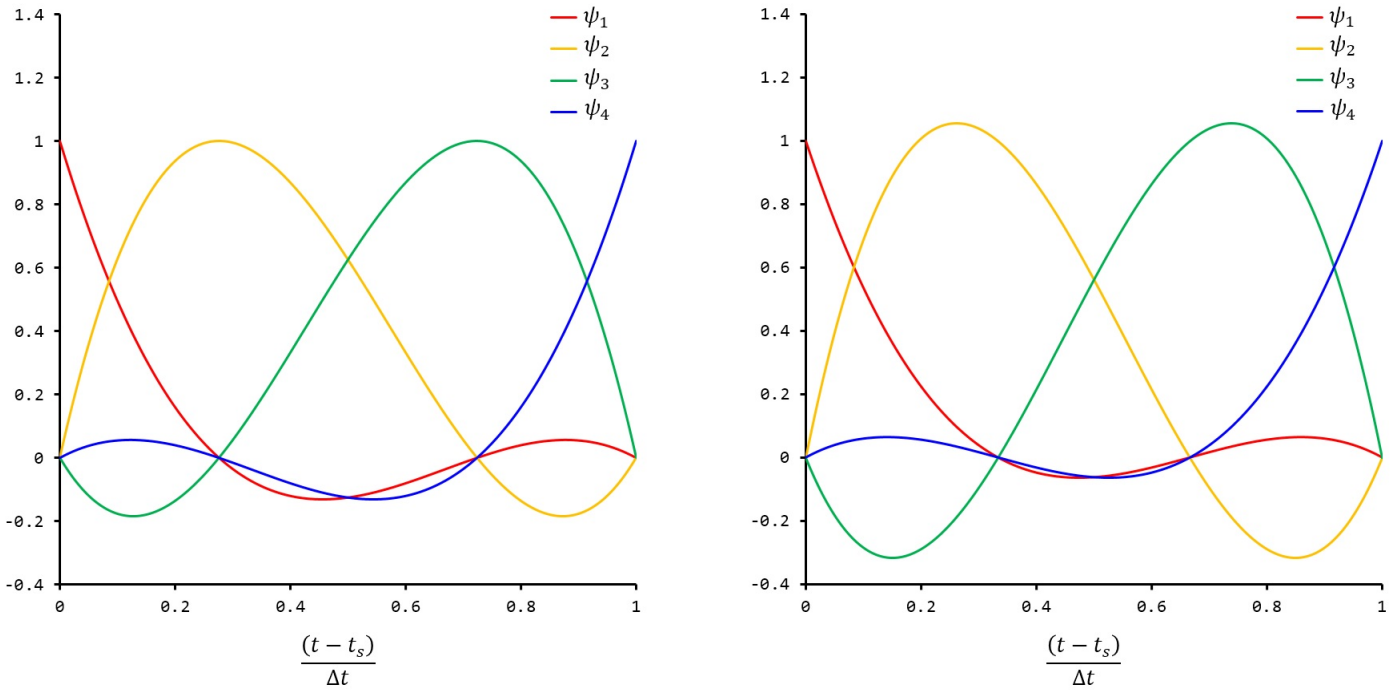

Figure 4.3: Comparison of 3rd-degree Lagrange interpolation functions. The left figure is the Gauss-Lobetto points based Lagrange interpolation functions, and the right figure is the equally space Lagrange interpolation functions.

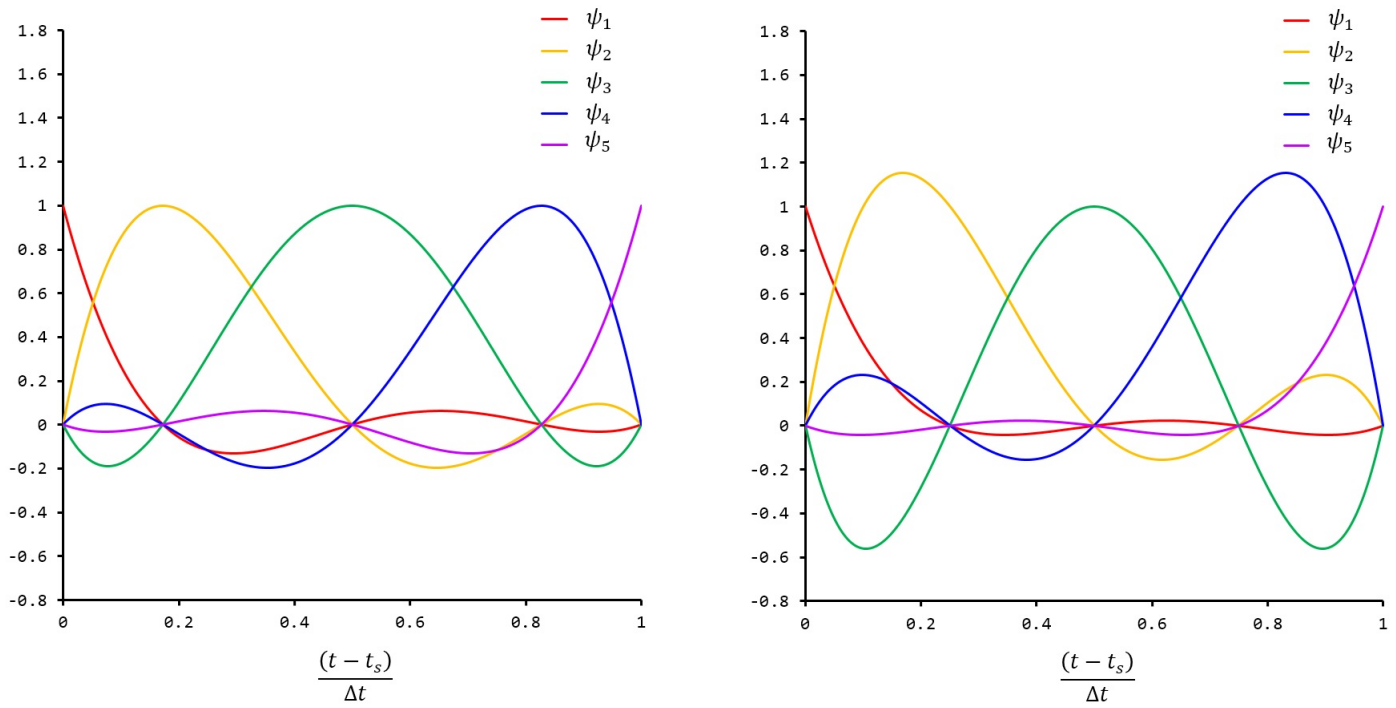

Figure 4.4: Comparison of 4th-degree Lagrange interpolation functions. The left figure is the Gauss-Lobetto points based Lagrange interpolation functions, and the right figure is the equally space Lagrange interpolation functions.

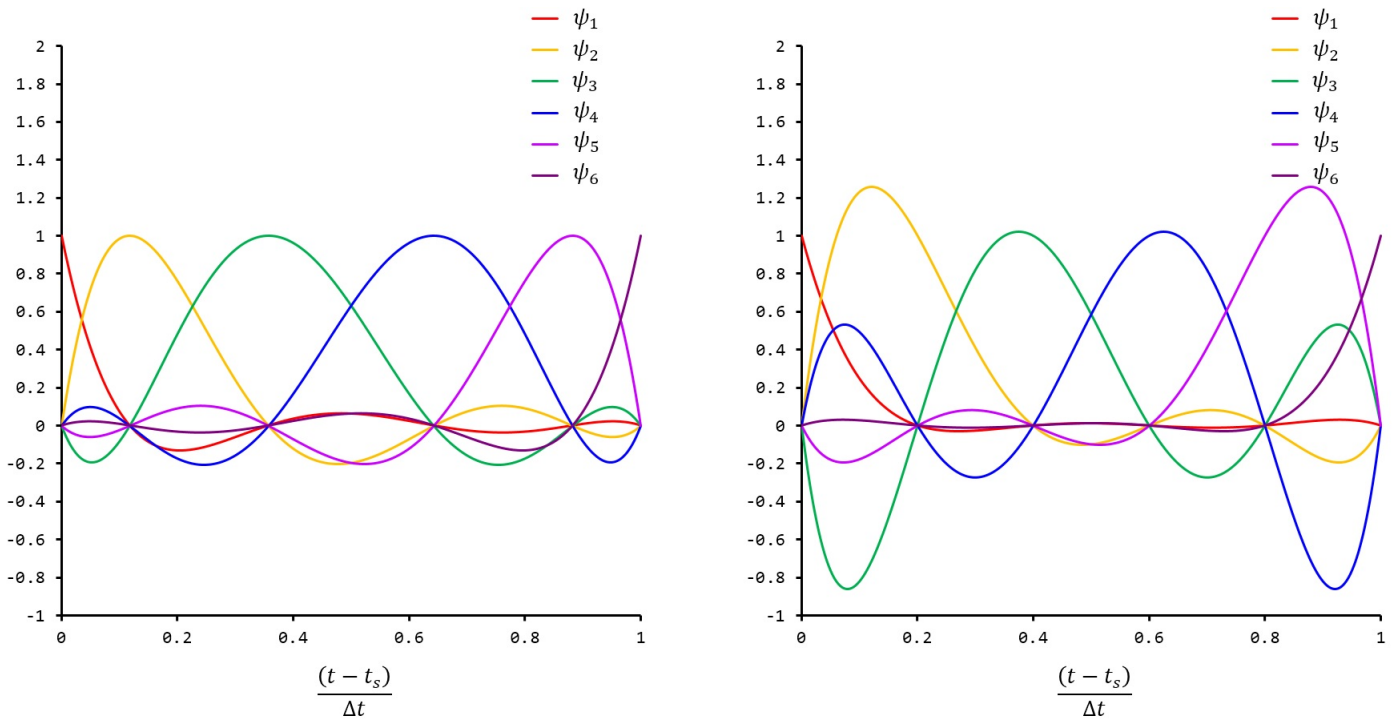

Figure 4.5: Comparison of 5th-degree Lagrange interpolation functions. The left figure is the Gauss-Lobetto points based Lagrange interpolation functions, and the right figure is the equally space Lagrange interpolation functions.

	Equally Spaced nodes	Gauss-Lobbato quadrature point based nodes
τ_0	0	0
τ_1	$\frac{1}{4}$	$-\frac{\sqrt{21}}{14}+\frac{1}{2}$
τ_2	$\frac{2}{4}$	$\frac{1}{2}$
τ_3	$\frac{3}{4}$	$\frac{\sqrt{21}}{14}+\frac{1}{2}$
τ_4	1	1

Table 4.2: Comparison of nodal spacing parameter τ_i for equally spaced and Gauss-Lobbato quadrature points based 4th-degree Lagrange interpolation functions

4.2.6 Final Form of Algorithms

If the nth-degree Lagrange functions are used for the approximations of time dependent variables (i.e., the displacement, velocity and acceleration vectors of the semi-discrete equation) in the proposed procedure, the nodal values of the 1st-order time derivative of the chosen time dependent variable can be expressed as the linear combination of n nodal values of the variable and two initial conditions. For the $(2n-1)$th-order accurate algorithm, the nodal velocities can be expressed as

$$\begin{Bmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_n \end{Bmatrix} = \begin{bmatrix} \alpha_{11}\mathbf{I} & \cdots & \alpha_{1n}\mathbf{I} \\ \vdots & \vdots & \vdots \\ \alpha_{n1}\mathbf{I} & \cdots & \alpha_{nn}\mathbf{I} \end{bmatrix} \begin{Bmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_n \end{Bmatrix} + \begin{Bmatrix} \beta_1\mathbf{I} \\ \vdots \\ \beta_n\mathbf{I} \end{Bmatrix} \mathbf{u}_0 + \begin{Bmatrix} \gamma_1\mathbf{I} \\ \vdots \\ \gamma_n\mathbf{I} \end{Bmatrix} \mathbf{v}_0 \tag{4.32}$$

where $\mathbf{u}_0$ and $\mathbf{v}_0$ are the initial displacement and velocity vectors; α_{ij}, β_i and γ_i are the algorithmic coefficients which can be directly defined as linear functions of μ. It can be observed that the computational structure of the final result given in Eq. (4.32) is very similar to the result equation of the differential quadrature method presented in Eq. (1.22).

The nodal accelerations can be also stated in terms of the nodal velocities and the initial conditions. The nodal accelerations can be expressed as

$$\begin{Bmatrix} \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_n \end{Bmatrix} = \begin{bmatrix} \alpha_{11}\mathbf{I} & \cdots & \alpha_{1n}\mathbf{I} \\ \vdots & \vdots & \vdots \\ \alpha_{n1}\mathbf{I} & \cdots & \alpha_{nn}\mathbf{I} \end{bmatrix} \begin{Bmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_n \end{Bmatrix} + \begin{Bmatrix} \beta_1\mathbf{I} \\ \vdots \\ \beta_n\mathbf{I} \end{Bmatrix} \mathbf{v}_0 + \begin{Bmatrix} \gamma_1\mathbf{I} \\ \vdots \\ \gamma_n\mathbf{I} \end{Bmatrix} \mathbf{a}_0 \tag{4.33}$$

where $\mathbf{a}_0$ is the initial acceleration vector. By using Eq. (4.32), the nodal accelerations given in Eq. (4.33) can be restated in terms of nodal displacements and the initial conditions as

$$\begin{Bmatrix} \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_n \end{Bmatrix} = \begin{bmatrix} \bar{\alpha}_{11}\mathbf{I} & \cdots & \bar{\alpha}_{1n}\mathbf{I} \\ \vdots & \vdots & \vdots \\ \bar{\alpha}_{n1}\mathbf{I} & \cdots & \bar{\alpha}_{nn}\mathbf{I} \end{bmatrix} \begin{Bmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_n \end{Bmatrix} + \begin{bmatrix} \bar{\beta}_1\mathbf{I} \\ \vdots \\ \bar{\beta}_n\mathbf{I} \end{bmatrix} \mathbf{u}_0 + \begin{bmatrix} (\bar{\gamma}_1 + \beta_1)\mathbf{I} \\ \vdots \\ (\bar{\gamma}_n + \beta_n)\mathbf{I} \end{bmatrix} \mathbf{v}_0 + \begin{bmatrix} \gamma_1\mathbf{I} \\ \vdots \\ \gamma_n\mathbf{I} \end{bmatrix} \mathbf{a}_0 \tag{4.34}$$

where, $\bar{\alpha}_{ij}$, $\bar{\beta}_i$, and $\bar{\gamma}_i$ are computed by using α_{ij}, β_i, and γ_i. $\bar{\alpha}_{ij}$, $\bar{\beta}_i$, and $\bar{\gamma}_i$ are

$$\begin{bmatrix} \bar{\alpha}_{11}\mathbf{I} & \cdots & \bar{\alpha}_{1n}\mathbf{I} \\ \vdots & \vdots & \vdots \\ \bar{\alpha}_{n1}\mathbf{I} & \cdots & \bar{\alpha}_{nn}\mathbf{I} \end{bmatrix} = \begin{bmatrix} \alpha_{11}\mathbf{I} & \cdots & \alpha_{1n}\mathbf{I} \\ \vdots & \vdots & \vdots \\ \alpha_{n1}\mathbf{I} & \cdots & \alpha_{nn}\mathbf{I} \end{bmatrix} \begin{bmatrix} \alpha_{11}\mathbf{I} & \cdots & \alpha_{1n}\mathbf{I} \\ \vdots & \vdots & \vdots \\ \alpha_{n1}\mathbf{I} & \cdots & \alpha_{nn}\mathbf{I} \end{bmatrix} \tag{4.35a}$$

$$\begin{bmatrix} \bar{\beta}_1\mathbf{I} \\ \vdots \\ \bar{\beta}_n\mathbf{I} \end{bmatrix} = \begin{bmatrix} \alpha_{11}\mathbf{I} & \cdots & \alpha_{1n}\mathbf{I} \\ \vdots & & \vdots \\ \alpha_{n1}\mathbf{I} & \cdots & \alpha_{nn}\mathbf{I} \end{bmatrix} \begin{bmatrix} \beta_1\mathbf{I} \\ \vdots \\ \beta_n\mathbf{I} \end{bmatrix} \tag{4.35b}$$

$$\begin{bmatrix} \bar{\gamma}_1\mathbf{I} \\ \vdots \\ \bar{\gamma}_n\mathbf{I} \end{bmatrix} = \begin{bmatrix} \alpha_{11}\mathbf{I} & \cdots & \alpha_{1n}\mathbf{I} \\ \vdots & \vdots & \vdots \\ \alpha_{n1}\mathbf{I} & \cdots & \alpha_{nn}\mathbf{I} \end{bmatrix} \begin{bmatrix} \gamma_1\ \mathbf{I} \\ \vdots \\ \gamma_n\ \mathbf{I} \end{bmatrix} \tag{4.35c}$$

4.2.6.1 Algorithms with the Equally Spaced Lagrange Interpolation Functions

Here, α_{ij}, β_i, and γ_i of the 1st, 3rd-, 5th-, 7th-, and 9th-order algorithms obtained from the equally spaced lagrange interpolation functions are presented. If $\mu = 1$ is used, each algorithm can provide one order higher accuracy (i.e., the $(2n)$th-order accuracy, n being the degree of approximations).

For the 1st-order accurate algorithm, α_{ij}, β_i, and γ_i are defined as

$$[\alpha_{ij}] = \frac{1}{\Delta t}\left[1+\mu\right], \quad \{\beta_i\} = \frac{1}{\Delta t}\left\{-1-\mu\right\}, \quad \{\gamma_i\} = \left\{-\mu\right\} \tag{4.36}$$

For the 3rd-order accurate algorithm, α_{ij}, β_i, and γ_i are defined as

$$\begin{aligned} [\alpha_{ij}] &= \frac{1}{\Delta t}\begin{bmatrix} \mu+1 & -\frac{1}{4}\mu+\frac{3}{4} \\ -4\mu-4 & \mu+3 \end{bmatrix} \\ \{\beta_i\} = \frac{1}{\Delta t}\begin{Bmatrix} -\frac{3}{4}\mu-\frac{7}{4} \\ 3\mu+1 \end{Bmatrix}, &\quad \{\gamma_i\} = \begin{Bmatrix} -\frac{1}{4}\mu-\frac{1}{4} \\ \mu \end{Bmatrix} \end{aligned} \tag{4.37}$$

For the 5th-order accurate algorithm, α_{ij}, β_i, and γ_i are defined as

$$
\begin{aligned}
[\alpha_{ij}] &= \frac{1}{\Delta t}\begin{bmatrix} \frac{1}{3}\mu+\frac{11}{6} & -\frac{1}{6}\mu+\frac{4}{3} & \frac{1}{27}\mu-\frac{7}{54} \\ -\frac{10}{3}\mu-\frac{10}{3} & \frac{5}{3}\mu+\frac{5}{3} & -\frac{10}{27}\mu+\frac{26}{27} \\ 9\mu+\frac{9}{2} & -\frac{9}{2}\mu-9 & \mu+\frac{11}{2} \end{bmatrix} \\
\{\beta_i\} &= \frac{1}{\Delta t}\begin{Bmatrix} -\frac{11}{54}\mu-\frac{82}{27} \\ \frac{55}{27}\mu+\frac{19}{27} \\ -\frac{11}{2}\mu-1 \end{Bmatrix}, \quad \{\gamma_i\} = \begin{Bmatrix} -\frac{1}{27}\mu-\frac{10}{27} \\ \frac{10}{27}\mu+\frac{1}{27} \\ -\mu \end{Bmatrix}
\end{aligned}
\tag{4.38}
$$

For the 7h-order accurate algorithm, α_{ij}, β_i, and γ_i are defined as

$$
\begin{aligned}
[\alpha_{ij}] &= \frac{1}{\Delta t}\begin{bmatrix} -\frac{19}{16}\mu+\frac{119}{48} & \frac{57}{64}\mu+\frac{105}{64} & -\frac{19}{48}\mu-\frac{1}{16} & \frac{19}{256}\mu-\frac{23}{768} \\ -3\mu-\frac{17}{3} & \frac{9}{4}\mu+\frac{9}{4} & -\mu+\frac{5}{3} & \frac{3}{16}\mu-\frac{7}{48} \\ \frac{93}{16}\mu+\frac{13}{16} & -\frac{279}{64}\mu-\frac{327}{64} & \frac{31}{16}\mu+\frac{47}{16} & -\frac{93}{256}\mu+\frac{275}{256} \\ -16\mu-\frac{16}{3} & 12\mu+12 & -\frac{16}{3}\mu-16 & \mu+\frac{25}{3} \end{bmatrix} \\
\{\beta_i\} &= \frac{1}{\Delta t}\begin{Bmatrix} \frac{475}{768}\mu-\frac{1031}{256} \\ \frac{25}{16}\mu+\frac{91}{48} \\ -\frac{775}{256}\mu+\frac{73}{256} \\ \frac{25}{3}\mu+1 \end{Bmatrix}, \quad \{\gamma_i\} = \begin{Bmatrix} \frac{19}{256}\mu-\frac{93}{256} \\ \frac{3}{16}\mu+\frac{3}{16} \\ -\frac{93}{256}\mu+\frac{19}{256} \\ \mu \end{Bmatrix}
\end{aligned}
\tag{4.39}
$$

For the 9th-order accurate algorithm, α_{ij}, β_i, and γ_i are defined as

$$[\alpha_{ij}] = \frac{1}{\Delta t}\begin{bmatrix} -\frac{399}{125}\mu + \frac{2387}{1500} & \frac{399}{125}\mu + \frac{374}{125} & -\frac{266}{125}\mu - \frac{41}{125} & \frac{399}{500}\mu - \frac{32}{375} & -\frac{399}{3125}\mu + \frac{379}{12500} \\ -\frac{118}{125}\mu - \frac{2311}{250} & \frac{118}{125}\mu + \frac{1904}{375} & -\frac{236}{375}\mu + \frac{63}{125} & \frac{59}{250}\mu + \frac{109}{250} & -\frac{118}{3125}\mu - \frac{1933}{18750} \\ \frac{843}{125}\mu + \frac{1097}{500} & -\frac{843}{125}\mu - \frac{743}{125} & \frac{562}{125}\mu + \frac{287}{125} & -\frac{843}{500}\mu + \frac{283}{125} & \frac{843}{3125}\mu - \frac{2653}{12500} \\ -\frac{876}{125}\mu + \frac{572}{375} & \frac{876}{125}\mu + \frac{226}{125} & -\frac{584}{125}\mu - \frac{984}{125} & \frac{219}{125}\mu + \frac{1732}{375} & -\frac{876}{3125}\mu + \frac{3524}{3125} \\ 25\mu + \frac{25}{4} & -25\mu - \frac{50}{3} & \frac{50}{3}\mu + 25 & -\frac{25}{4}\mu - 25 & \mu + \frac{137}{12} \end{bmatrix}$$

$$\{\beta_i\} = \frac{1}{\Delta t}\begin{Bmatrix} \frac{18221}{12500}\mu - \frac{13126}{3125} \\ \frac{8083}{18750}\mu + \frac{20811}{6250} \\ -\frac{38497}{12500}\mu - \frac{1868}{3125} \\ \frac{10001}{3125}\mu - \frac{3774}{3125} \\ -\frac{137}{12}\mu - 1 \end{Bmatrix}, \quad \{\gamma_i\} = \begin{Bmatrix} \frac{399}{3125}\mu - \frac{876}{3125} \\ \frac{118}{3125}\mu + \frac{843}{3125} \\ -\frac{843}{3125}\mu - \frac{118}{3125} \\ \frac{876}{3125}\mu - \frac{399}{3125} \\ -\mu \end{Bmatrix} \tag{4.40}$$

4.2.6.2 Algorithms with the Gauss-Lobatto Point Based Lagrange Interpolation Functions

Here, α_{ij}, β_i, and γ_i of the 5th-, 7th- and 9th-order accurate algorithms obtained from the Gauss-Lobatto quadrature points based Lagrange interpolation functions are presented. Again, each algorithm can provide one order higher accuracy for the choice of $\mu = 1$.

For the 5th-order accurate algorithm, α_{ij}, β_i, and γ_i are defined as

$$
\begin{aligned}
[\alpha_{ij}] = \frac{1}{\Delta t} \Bigg(\mu & \begin{bmatrix} 1 & -\frac{3}{2}+\frac{1}{2}\sqrt{5} & -\frac{1}{10}+\frac{1}{10}\sqrt{5} \\ -\frac{3}{2}-\frac{1}{2}\sqrt{5} & 1 & -\frac{1}{10}-\frac{1}{10}\sqrt{5} \\ \frac{5}{2}+\frac{5}{2}\sqrt{5} & \frac{5}{2}-\frac{5}{2}\sqrt{5} & 1 \end{bmatrix} \\
& + \begin{bmatrix} \frac{3}{2}+\frac{1}{2}\sqrt{5} & -1+\sqrt{5} & \frac{3}{5}-\frac{2}{5}\sqrt{5} \\ -1-\sqrt{5} & \frac{3}{2}-\frac{1}{2}\sqrt{5} & \frac{3}{5}+\frac{2}{5}\sqrt{5} \\ -\frac{5}{2}+\frac{5}{2}\sqrt{5} & -\frac{5}{2}-\frac{5}{2}\sqrt{5} & 6 \end{bmatrix} \Bigg) \\
\{\beta_i\} = \frac{1}{\Delta t} & \begin{Bmatrix} \frac{3}{5}\,\mu - \frac{3}{5}\sqrt{5}\mu - \frac{11}{10} - \frac{11}{10}\sqrt{5} \\ \frac{3}{5}\,\mu + \frac{3}{5}\sqrt{5}\mu - \frac{11}{10} + \frac{11}{10}\sqrt{5} \\ -6\,\mu - 1 \end{Bmatrix} \\
\{\gamma_i\} = & \begin{Bmatrix} \frac{1}{10}\,\mu - \frac{1}{10}\sqrt{5}\mu - \frac{1}{10} - \frac{1}{10}\sqrt{5} \\ \frac{1}{10}\,\mu + \frac{1}{10}\sqrt{5}\mu - \frac{1}{10} + \frac{1}{10}\sqrt{5} \\ -\mu \end{Bmatrix}
\end{aligned}
\tag{4.41}
$$

For the 7th-order accurate algorithm, α_{ij}, β_i, and γ_i are defined as

$$[\alpha_{ij}] = \frac{1}{\Delta t}\left(\mu \begin{bmatrix} 1 & -\frac{8}{7}+\frac{8}{49}\sqrt{21} & \frac{5}{2}-\frac{1}{2}\sqrt{21} & -\frac{3}{14}+\frac{3}{98}\sqrt{21} \\ -\frac{7}{32}\sqrt{21}-\frac{49}{32} & 1 & -\frac{49}{32}+\frac{7}{32}\sqrt{21} & \frac{3}{16} \\ \frac{5}{2}+\frac{1}{2}\sqrt{21} & -\frac{8}{49}\sqrt{21}-\frac{8}{7} & 1 & -\frac{3}{98}\sqrt{21}-\frac{3}{14} \\ -\frac{7}{6}\sqrt{21}-\frac{49}{6} & \frac{16}{3} & -\frac{49}{6}+\frac{7}{6}\sqrt{21} & 1 \end{bmatrix} \right.$$
$$\left. + \begin{bmatrix} \frac{5}{2}+\frac{1}{2}\sqrt{21} & -\frac{8}{7}+\frac{88}{147}\sqrt{21} & 1-\frac{1}{3}\sqrt{21} & \frac{9}{7}-\frac{12}{49}\sqrt{21} \\ -\frac{49}{32}-\frac{77}{96}\sqrt{21} & 1 & -\frac{49}{32}+\frac{77}{96}\sqrt{21} & -\frac{9}{16} \\ 1+\frac{1}{3}\sqrt{21} & -\frac{8}{7}-\frac{88}{147}\sqrt{21} & \frac{5}{2}-\frac{1}{2}\sqrt{21} & \frac{12}{49}\sqrt{21}+\frac{9}{7} \\ -\frac{49}{6}+\frac{7}{6}\sqrt{21} & \frac{16}{3} & -\frac{7}{6}\sqrt{21}-\frac{49}{6} & 10 \end{bmatrix} \right)$$

$$\{\beta_i\} = \frac{1}{\Delta t}\begin{Bmatrix} -\frac{15}{7}\,\mu+\frac{15}{49}\sqrt{21}\mu-\frac{51}{14}-\frac{51}{98}\sqrt{21} \\ \frac{15}{8}\,\mu+\frac{21}{8} \\ -\frac{15}{49}\sqrt{21}\mu-\frac{15}{7}\,\mu+\frac{51}{98}\sqrt{21}-\frac{51}{14} \\ 10\,\mu+1 \end{Bmatrix}$$

$$\{\gamma_i\} = \begin{Bmatrix} -\frac{3}{14}\,\mu+\frac{3}{98}\sqrt{21}\mu-\frac{3}{14}-\frac{3}{98}\sqrt{21} \\ \frac{3}{16}\,\mu+\frac{3}{16} \\ -\frac{3}{98}\sqrt{21}\mu-\frac{3}{14}\,\mu+\frac{3}{98}\sqrt{21}-\frac{3}{14} \\ \mu \end{Bmatrix} \tag{4.42}$$

For the 9th-order accurate algorithm, α_{ij}, β_i, and γ_i are defined as

$$[\alpha_{ij}] = \frac{1}{\Delta t}\left(\mu \begin{bmatrix} 1.0 & -0.3979905070 & 0.2213383872 & -0.1331089586 & 0.04930278012 \\ -2.512622744 & 1.0 & -0.5561398661 & 0.3344525970 & -0.1238792867 \\ 4.517969126 & -1.798108823 & 1.0 & -0.6013821654 & 0.2227484384 \\ -7.512642352 & 2.989960338 & -1.662836143 & 1.0 & -0.3703941540 \\ 20.28283188 & -8.072374540 & 4.489369296 & -2.699826628 & 1.0 \end{bmatrix} \right.$$
$$\left. + \begin{bmatrix} 7.512642350 & 2.056893217 & -0.6428201742 & 0.3070950148 & -0.1051682020 \\ -7.960483029 & 1.798108823 & 2.505923933 & -0.9713311794 & 0.3166527833 \\ 4.085336090 & -4.505923934 & 0.5561398660 & 3.108061311 & -0.8460228092 \\ -2.307095015 & 2.703646823 & -5.268191942 & 0.1331089586 & 3.523427116 \\ 2.699826629 & -4.489369295 & 8.072374541 & -20.28283186 & 15.0 \end{bmatrix} \right) \tag{4.43}$$

$$\{\beta_i\} = \frac{1}{\Delta t}\begin{Bmatrix} -0.73954170180\,\mu - 9.128642207 \\ 1.85818930050\,\mu + 4.311128671 \\ -3.3412265760\,\mu - 2.397590522 \\ 5.5559123100\,\mu + 1.215104058 \\ -15.0\,\mu - 1.0 \end{Bmatrix}, \quad \{\gamma_i\} = \begin{Bmatrix} -0.04930278012\,\mu - 0.3703941540 \\ 0.1238792867\,\mu + 0.2227484384 \\ -0.2227484384\,\mu - 0.1238792867 \\ 0.3703941540\,\mu + 0.04930278012 \\ -\mu \end{Bmatrix}$$

4.2.7 Linear Equation Solving Procedure

To find n unknown displacement vectors (i.e., $\mathbf{u}_1, ..., \mathbf{u}_n$), we need n dynamic equilibrium equations, and they can be obtained by evaluating Eq. (4.3a) at $t_i = t_s + \tau_i\ \Delta t$ (for $i = 1, ..., n$). These n equilibrium equations can be written as the matrix form of

$$\begin{bmatrix} \mathbf{M} & & \\ & \ddots & \\ & & \mathbf{M} \end{bmatrix} \begin{Bmatrix} \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_n \end{Bmatrix} + \begin{bmatrix} \mathbf{C} & & \\ & \ddots & \\ & & \mathbf{C} \end{bmatrix} \begin{Bmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_n \end{Bmatrix} + \begin{bmatrix} \mathbf{K} & & \\ & \ddots & \\ & & \mathbf{K} \end{bmatrix} \begin{Bmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_n \end{Bmatrix} = \begin{Bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_n \end{Bmatrix} \tag{4.44}$$

where, $\mathbf{a}_i$, $\mathbf{v}_i$, $\mathbf{u}_i$, and $\mathbf{f}_i$ are the ith time nodal acceleration, velocity, displacement, and external force vectors, respectively. Then by substituting Eqs. (4.32) and (4.34) into Eq. (4.44) we can obtain

$$\begin{aligned}&\left(\begin{bmatrix}\bar{\alpha}_{11}\mathbf{M} & \cdots & \bar{\alpha}_{1n}\mathbf{M}\\ \vdots & \vdots & \vdots\\ \bar{\alpha}_{n1}\mathbf{M} & \cdots & \bar{\alpha}_{nn}\mathbf{M}\end{bmatrix}+\begin{bmatrix}\alpha_{11}\mathbf{C} & \cdots & \alpha_{1n}\mathbf{C}\\ \vdots & \vdots & \vdots\\ \alpha_{n1}\mathbf{C} & \cdots & \alpha_{nn}\mathbf{C}\end{bmatrix}+\begin{bmatrix}\mathbf{K} & & \\ & \ddots & \\ & & \mathbf{K}\end{bmatrix}\right)\begin{Bmatrix}\mathbf{u}_1\\ \vdots\\ \mathbf{u}_n\end{Bmatrix}\\ &=\begin{Bmatrix}\mathbf{f}_1\\ \vdots\\ \mathbf{f}_n\end{Bmatrix}-\begin{bmatrix}\bar{\beta}_1\mathbf{M}+\beta_1\mathbf{C}\\ \vdots\\ \bar{\beta}_n\mathbf{M}+\beta_\mathbf{n}\mathbf{C}\end{bmatrix}\mathbf{u}_0-\begin{bmatrix}(\bar{\gamma}_1+\beta_1)\mathbf{M}+\gamma_1\mathbf{C}\\ \vdots\\ (\bar{\gamma}_n+\beta_n)\mathbf{M}+\gamma_n\mathbf{C}\end{bmatrix}\mathbf{v}_0-\begin{bmatrix}\gamma_1\mathbf{M}\\ \vdots\\ \gamma_n\mathbf{M}\end{bmatrix}\mathbf{a}_0\end{aligned} \tag{4.45}$$

Now Eq. (4.45) can be solved to find $\mathbf{u}_1, ..., \mathbf{u}_n$. However, $\mathbf{a}_0$ should be computed as $\mathbf{a}_0 = \mathbf{M}^{-1}(\mathbf{f}_0 - \mathbf{C}\mathbf{v}_0 - \mathbf{K}\mathbf{u}_0)$ at $t = 0$. This initial computation of $\mathbf{a}_0$ can be avoided by utilizing the dynamic equilibrium equation at $t = t_0$ (i.e., $\mathbf{M}\mathbf{a}_0 + \mathbf{C}\mathbf{v}_0 + \mathbf{K}\mathbf{u}_0 = \mathbf{f}_0$). Since the expression of $-\gamma_i\mathbf{M}\mathbf{a}_0 - \gamma_i\mathbf{C}\mathbf{v}_0$ in Eq. (4.45) can be replaced by the expression of $\gamma_i\mathbf{K}\mathbf{u}_0 - \gamma_i\mathbf{f}_0$, and we can finally obtain

$$\begin{aligned}&\left(\begin{bmatrix}\bar{\alpha}_{11}\mathbf{M} & \cdots & \bar{\alpha}_{1n}\mathbf{M}\\ \vdots & \vdots & \vdots\\ \bar{\alpha}_{n1}\mathbf{M} & \cdots & \bar{\alpha}_{nn}\mathbf{M}\end{bmatrix}+\begin{bmatrix}\alpha_{11}\mathbf{C} & \cdots & \alpha_{1n}\mathbf{C}\\ \vdots & \vdots & \vdots\\ \alpha_{n1}\mathbf{C} & \cdots & \alpha_{nn}\mathbf{C}\end{bmatrix}+\begin{bmatrix}\mathbf{K} & & \\ & \ddots & \\ & & \mathbf{K}\end{bmatrix}\right)\begin{Bmatrix}\mathbf{u}_1\\ \vdots\\ \mathbf{u}_n\end{Bmatrix}\\ &=\begin{Bmatrix}\mathbf{f}_1\\ \vdots\\ \mathbf{f}_n\end{Bmatrix}-\begin{bmatrix}\bar{\beta}_1\mathbf{M}+\beta_1\mathbf{C}-\gamma_1\mathbf{K}\\ \vdots\\ \bar{\beta}_n\mathbf{M}+\beta_\mathbf{n}\mathbf{C}-\gamma_n\mathbf{K}\end{bmatrix}\mathbf{u}_0-\begin{bmatrix}(\bar{\gamma}_1+\beta_1)\mathbf{M}\\ \vdots\\ (\bar{\gamma}_n+\beta_1)\mathbf{M}\end{bmatrix}\mathbf{v}_0-\begin{bmatrix}\gamma_1\mathbf{I}\\ \vdots\\ \gamma_n\mathbf{I}\end{bmatrix}\mathbf{f}_0\end{aligned} \tag{4.46}$$

Here, α_{ij}, β_i, and γ_i can be computed by using the results given in Eqs. (4.36)-(4.42). After finding $\mathbf{u}_1, ..., \mathbf{u}_n$ by solving Eq. (4.45) or Eq. (4.46), the velocity and acceleration vectors can also be updated by using Eqs. (4.32) and (4.34). After

obtaining $\mathbf{u}_n$, $\mathbf{v}_n$, and $\mathbf{a}_n$, the initial conditions $\mathbf{u}_0$, $\mathbf{v}_0$, and $\mathbf{a}_0$ of Eqs. (4.45), (4.46), (4.32) and (4.34) should be updated as $\mathbf{u}_n$, $\mathbf{v}_n$, and $\mathbf{a}_n$ to advance another step.

4.2.8 Nonlinear Equation Solving Procedures

As stated previously, the biggest advantage of the differential quadrature method is direct and intuitive extensibility to nonlinear cases and other types of initial value problems. Since the computational structures of the result equations obtained from the proposed procedure are very similar to those of the differential quadrature method based algorithms as presented in Eqs. (4.32) and (1.22), we can analyse nonlinear structural dynamics problems and other types of initial value problems by using the newly developed algorithms without any difficulty. In a general form, the nonlinear equation of structural dynamics can be written as

$$\mathbf{M}(t)\ddot{\mathbf{u}}(t) + \mathbf{C}(t)\dot{\mathbf{u}}(t) + \mathbf{n}(t) = \mathbf{f}(t) \tag{4.47}$$

with the initial conditions

$$\mathbf{u}(0) = \mathbf{u}_0 \tag{4.48a}$$

$$\dot{\mathbf{u}}(0) = \mathbf{v}_0 \tag{4.48b}$$

where, $\mathbf{M}(t)$ and $\mathbf{C}(t)$ are the mass and damping matrices; $\mathbf{n}(t)$ is the nonlinear internal stress related force vector. Note that we are not assuming that the mass and damping matrices are constants, but they are allowed to contain certain nonlinearities.

4.2.8.1 Direct Iterative Method

The simplest nonlinear equation solving method is the direct iterative method [4]. In many cases, $\mathbf{n}(t)$ can be linearized as $\mathbf{K}(t)\mathbf{u}(t)$, where $\mathbf{K}(t)$ is the linearized nonlinear stiffness matrix. For the nth-degree approximations (i.e., the $(2n-1)$th-order accurate algorithm), n nonlinear dynamic equilibrium equations at $t_s + \tau_1 \Delta t, ..., t_s + \tau_n \Delta t$ can be written as

$$\begin{bmatrix} \mathbf{M}_1 & & \\ & \ddots & \\ & & \mathbf{M}_n \end{bmatrix} \begin{Bmatrix} \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_n \end{Bmatrix} + \begin{bmatrix} \mathbf{C}_1 & & \\ & \ddots & \\ & & \mathbf{C}_n \end{bmatrix} \begin{Bmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_n \end{Bmatrix} + \begin{bmatrix} \mathbf{K}_1 & & \\ & \ddots & \\ & & \mathbf{K}_n \end{bmatrix} \begin{Bmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_n \end{Bmatrix} = \begin{Bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_n \end{Bmatrix} \tag{4.49}$$

where, $\mathbf{M}_i$, $\mathbf{C}_i$, and $\mathbf{K}_i$ are the mass, damping, and stiffness matrices evaluated at $t_i = t_s + \tau_i \Delta t$ for $i = 1, ..., n$. After proper linearizations of nonlinearities, it is assumed that $\mathbf{M}_i$, $\mathbf{C}_i$, and $\mathbf{K}_i$ can be computed by using the rth iterative solution at ith time node. By substituting the nodal acceleration and velocity vectors given in Eqs. (4.34) and (4.32) into Eq. (4.49), we obtain

$$\begin{aligned} &\left(\begin{bmatrix} \bar{\alpha}_{11}\mathbf{M}_1^{(r)} & \cdots & \bar{\alpha}_{1n}\mathbf{M}_1^{(r)} \\ \vdots & \vdots & \vdots \\ \bar{\alpha}_{n1}\mathbf{M}_n^{(r)} & \cdots & \bar{\alpha}_{nn}\mathbf{M}_n^{(r)} \end{bmatrix} + \begin{bmatrix} \alpha_{11}\mathbf{C}_1^{(r)} & \cdots & \alpha_{1n}\mathbf{C}_1^{(r)} \\ \vdots & \vdots & \vdots \\ \alpha_{n1}\mathbf{C}_n^{(r)} & \cdots & \alpha_{nn}\mathbf{C}_n^{(r)} \end{bmatrix} + \begin{bmatrix} \mathbf{K}_1^{(r)} & & \\ & \ddots & \\ & & \mathbf{K}_n^{(r)} \end{bmatrix} \right) \begin{Bmatrix} \mathbf{u}_1^{(r+1)} \\ \vdots \\ \mathbf{u}_n^{(r+1)} \end{Bmatrix} \\ &= \begin{Bmatrix} \mathbf{f}_1^{(r)} \\ \vdots \\ \mathbf{f}_n^{(r)} \end{Bmatrix} - \begin{bmatrix} \bar{\beta}_1\mathbf{M}_1^{(r)} + \beta_1\mathbf{C}_1^{(r)} \\ \vdots \\ \bar{\beta}_n\mathbf{M}_n^{(r)} + \beta_n\mathbf{C}_n^{(r)} \end{bmatrix} \mathbf{u}_0 - \begin{bmatrix} (\bar{\gamma}_1 + \beta_1)\mathbf{M}_1^{(r)} + \gamma_1\mathbf{C}_1^{(r)} \\ \vdots \\ (\bar{\gamma}_n + \beta_1)\mathbf{M}_n^{(r)} + \gamma_n\mathbf{C}_n^{(r)} \end{bmatrix} \mathbf{v}_0 - \begin{bmatrix} \gamma_1\mathbf{M}_1^{(r)} \\ \vdots \\ \gamma_n\mathbf{M}_n^{(r)} \end{bmatrix} \mathbf{a}_0 \end{aligned} \tag{4.50}$$

where, properties with the superscript (r) denote that those properties are evaluated by using the rth iterative solution (i.e., $\mathbf{u}_1^{(r)}, ..., \mathbf{u}_n^{(r)}$). In the direct iterative method, the $(r+1)$th iterative solution $\mathbf{u}_1^{(r+1)}, ..., \mathbf{u}_n^{(r+1)}$ can be directly obtained by solving Eq. (4.50). For each time step, finding $\mathbf{u}_1^{(r+1)}, ..., \mathbf{u}_n^{(r+1)}$ and evaluating $\mathbf{M}_i^{(r)}$, $\mathbf{C}_i^{(r)}$, and $\mathbf{K}_i^{(r)}$ should be repeated until properly converged solutions are obtained. Since the computational structures of linear and nonlinear cases are almost the same, the direct iterative method can easily be implemented as the computer code by sharing the linear computer code, once a proper iteration loop is added.

4.2.8.2 Newton-Raphson Iterative Method

For the newly developed algorithms, the Newton-Raphson iterative method [4, 44] can also be used without any difficulty. In general, the Newton-Raphson method provides a much faster convergence rate than the direct method. In highly nonlinear problems, the Newton-Raphson method is recommended instead of the direct iterative method. To apply the Newton-Raphson method to the solving of nonlinear equations, we define a residual vector caused by the unconverged solutions as

$$\mathbf{r}(t) = \mathbf{M}(t)\ddot{\mathbf{u}}(t) + \mathbf{C}(t)\dot{\mathbf{u}}(t) + \mathbf{n}(t) - \mathbf{f}(t) \tag{4.51}$$

and the residual vectors evaluated at the time nodal points can be written as

$$\begin{Bmatrix} \mathbf{r}_1 \\ \vdots \\ \mathbf{r}_n \end{Bmatrix} = \begin{bmatrix} \mathbf{M}_1 & & \\ & \ddots & \\ & & \mathbf{M}_n \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{u}}_1 \\ \vdots \\ \ddot{\mathbf{u}}_n \end{Bmatrix} + \begin{bmatrix} \mathbf{C}_1 & & \\ & \ddots & \\ & & \mathbf{C}_n \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{u}}_1 \\ \vdots \\ \dot{\mathbf{u}}_n \end{Bmatrix} + \begin{Bmatrix} \mathbf{n}_1 \\ \vdots \\ \mathbf{n}_n \end{Bmatrix} - \begin{Bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_n \end{Bmatrix} \tag{4.52}$$

where $\mathbf{r}_1, ..., \mathbf{r}_n$ are the residual vectors, $\mathbf{r}_i$ being the residual vector associated with the ith time node at $t = t_s + \tau_i \Delta t$. The Newton-Raphson method can be applied

to Eq. (4.52) to find $\mathbf{u}_1, ..., \mathbf{u}_n$ which satisfies $\mathbf{r}_1, ..., \mathbf{r}_n = \{\mathbf{0}\}$. By using the Taylor expansion of $\mathbf{r}(t)$ defined in Eq. (4.51), the residual vectors of the $(r+1)$th iteration can be stated as

$$\begin{Bmatrix} \mathbf{r}_1^{(r+1)} \\ \vdots \\ \mathbf{r}_n^{(r+1)} \end{Bmatrix} = \begin{Bmatrix} \mathbf{r}_1^{(r)} \\ \vdots \\ \mathbf{r}_n^{(r)} \end{Bmatrix} + \begin{bmatrix} \frac{\partial}{\partial \mathbf{u}} \mathbf{r}(t)\big|_{\mathbf{u}=\mathbf{u}_1^{(r)}} & & \\ & \ddots & \\ & & \frac{\partial}{\partial \mathbf{u}} \mathbf{r}(t)\big|_{\mathbf{u}=\mathbf{u}_n^{(r)}} \end{bmatrix} \begin{Bmatrix} \Delta\mathbf{u}_1^{(r)} \\ \vdots \\ \Delta\mathbf{u}_n^{(r)} \end{Bmatrix} = \begin{Bmatrix} \mathbf{0} \\ \vdots \\ \mathbf{0} \end{Bmatrix} \tag{4.53}$$

where properties with the super script (r) and $(r+1)$ denote that they are evaluated by using the rth and $(r+1)$th iterative solutions, respectively. In the Newton-Raphson method, the ith nodal displacement solutions are updated as $\mathbf{u}_i^{(r+1)} = \mathbf{u}_i^{(r)} + \Delta\mathbf{u}_i^{(r)}$. From Eq. (4.53), the incremental equations can be written as

$$\begin{aligned} &\begin{bmatrix} \mathbf{M}_1^{(r)} & & \\ & \ddots & \\ & & \mathbf{M}_n^{(r)} \end{bmatrix} \begin{Bmatrix} \Delta\ddot{\mathbf{u}}_1^{(r)} \\ \vdots \\ \Delta\ddot{\mathbf{u}}_n^{(r)} \end{Bmatrix} + \begin{bmatrix} \mathbf{C}_1^{(r)} & & \\ & \ddots & \\ & & \mathbf{C}_n^{(r)} \end{bmatrix} \begin{Bmatrix} \Delta\dot{\mathbf{u}}_1^{(r)} \\ \vdots \\ \Delta\dot{\mathbf{u}}_n^{(r)} \end{Bmatrix} \\ &\qquad + \begin{bmatrix} \mathbf{T}_1^{(r)} & & \\ & \ddots & \\ & & \mathbf{T}_n^{(r)} \end{bmatrix} \begin{Bmatrix} \Delta\mathbf{u}_1^{(r)} \\ \vdots \\ \Delta\mathbf{u}_n^{(r)} \end{Bmatrix} = - \begin{Bmatrix} \mathbf{r}_1^{(r)} \\ \vdots \\ \mathbf{r}_n^{(r)} \end{Bmatrix} \end{aligned} \tag{4.54}$$

where the tangent matrix associated with ith time node can be computed as

$$\mathbf{T}_i^{(r)} = \left(\frac{\partial \mathbf{M}}{\partial \mathbf{u}} \ddot{\mathbf{u}} + \frac{\partial \mathbf{C}}{\partial \mathbf{u}} \dot{\mathbf{u}} + \frac{\partial \mathbf{n}}{\partial \mathbf{u}} - \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \right) \Bigg|_{\mathbf{u}=\mathbf{u}_i^{(r)}} \quad \text{for} \quad i = 1, 2, ..., n \tag{4.55}$$

Then, by using Eqs. (4.34) and (4.32), $\Delta\ddot{\mathbf{u}}_1^{(r)}, ..., \Delta\ddot{\mathbf{u}}_n^{(r)}$ and $\Delta\dot{\mathbf{u}}_1^{(r)}, ..., \Delta\dot{\mathbf{u}}_n^{(r)}$ in Eq. (4.54) can be stated as

$$\begin{Bmatrix} \Delta\dot{\mathbf{u}}_1^{(r)} \\ \vdots \\ \Delta\dot{\mathbf{u}}_n^{(r)} \end{Bmatrix} = \begin{Bmatrix} \mathbf{v}_1^{(r+1)} \\ \vdots \\ \mathbf{v}_n^{(r+1)} \end{Bmatrix} - \begin{Bmatrix} \mathbf{v}_1^{(r)} \\ \vdots \\ \mathbf{v}_n^{(r)} \end{Bmatrix} = \begin{bmatrix} \alpha_{11}\mathbf{I} & \cdots & \alpha_{1n}\mathbf{I} \\ \vdots & \vdots & \vdots \\ \alpha_{n1}\mathbf{I} & \cdots & \alpha_{nn}\mathbf{I} \end{bmatrix} \begin{Bmatrix} \Delta\mathbf{u}_1^{(r)} \\ \vdots \\ \Delta\mathbf{u}_n^{(r)} \end{Bmatrix} \tag{4.56a}$$

$$\begin{Bmatrix} \Delta\ddot{\mathbf{u}}_1^{(r)} \\ \vdots \\ \Delta\ddot{\mathbf{u}}_n^{(r)} \end{Bmatrix} = \begin{Bmatrix} \mathbf{a}_1^{(r+1)} \\ \vdots \\ \mathbf{a}_n^{(r+1)} \end{Bmatrix} - \begin{Bmatrix} \mathbf{a}_1^{(r)} \\ \vdots \\ \mathbf{a}_n^{(r)} \end{Bmatrix} = \begin{bmatrix} \bar{\alpha}_{11}\mathbf{I} & \cdots & \bar{\alpha}_{1n}\mathbf{I} \\ \vdots & \vdots & \vdots \\ \bar{\alpha}_{n1}\mathbf{I} & \cdots & \bar{\alpha}_{nn}\mathbf{I} \end{bmatrix} \begin{Bmatrix} \Delta\mathbf{u}_1^{(r)} \\ \vdots \\ \Delta\mathbf{u}_n^{(r)} \end{Bmatrix} \tag{4.56b}$$

and the incremental solutions $\Delta\mathbf{u}_1^{(r)}, ..., \Delta\mathbf{u}_n^{(r)}$ can be obtained by solving

$$\left(\begin{bmatrix} \bar{\alpha}_{11}\mathbf{M}_1^{(r)} \cdots \bar{\alpha}_{1n}\mathbf{M}_1^{(r)} \\ \vdots \quad \vdots \quad \vdots \\ \bar{\alpha}_{n1}\mathbf{M}_n^{(r)} \cdots \bar{\alpha}_{nn}\mathbf{M}_n^{(r)} \end{bmatrix} + \begin{bmatrix} \alpha_{11}\mathbf{C}_1^{(r)} \cdots \alpha_{1n}\mathbf{C}_1^{(r)} \\ \vdots \quad \vdots \quad \vdots \\ \alpha_{n1}\mathbf{C}_n^{(r)} \cdots \alpha_{nn}\mathbf{C}_n^{(r)} \end{bmatrix} + \begin{bmatrix} \mathbf{T}_1^{(r)} & & \\ & \ddots & \\ & & \mathbf{T}_n^{(r)} \end{bmatrix} \right) \begin{Bmatrix} \Delta\mathbf{u}_1^{(r)} \\ \vdots \\ \Delta\mathbf{u}_n^{(r)} \end{Bmatrix} = - \begin{Bmatrix} \mathbf{r}_1^{(r)} \\ \vdots \\ \mathbf{r}_n^{(r)} \end{Bmatrix} \tag{4.57}$$

After obtaining the incremental solutions $\Delta\mathbf{u}_1^{(r)}, ..., \Delta\mathbf{u}_n^{(r)}$ by solving Eq. (4.57), $\mathbf{u}_1^{(r+1)}, ..., \mathbf{u}_n^{(r+1)}$ is updated as

$$\begin{Bmatrix} \mathbf{u}_1^{(r+1)} \\ \vdots \\ \mathbf{u}_n^{(r+1)} \end{Bmatrix} = \begin{Bmatrix} \mathbf{u}_1^{(r)} \\ \vdots \\ \mathbf{u}_n^{(r)} \end{Bmatrix} + \begin{Bmatrix} \Delta\mathbf{u}_1^{(r)} \\ \vdots \\ \Delta\mathbf{u}_n^{(r)} \end{Bmatrix} \tag{4.58}$$

Then $\mathbf{v}_1^{(r+1)}, ..., \mathbf{v}_n^{(r+1)}$ and $\mathbf{a}_1^{(r+1)}, ..., \mathbf{a}_n^{(r+1)}$ are also updated as

$$\begin{Bmatrix} \mathbf{v}_1^{(r+1)} \\ \vdots \\ \mathbf{v}_n^{(r+1)} \end{Bmatrix} = \begin{bmatrix} \alpha_{11}\mathbf{I} & \cdots & \alpha_{1n}\mathbf{I} \\ \vdots & \vdots & \vdots \\ \alpha_{n1}\mathbf{I} & \cdots & \alpha_{nn}\mathbf{I} \end{bmatrix} \begin{Bmatrix} \mathbf{u}_1^{(r+1)} \\ \vdots \\ \mathbf{u}_n^{(r+1)} \end{Bmatrix} + \begin{Bmatrix} \beta_1\mathbf{I} \\ \vdots \\ \beta_n\mathbf{I} \end{Bmatrix} \mathbf{u}_0 + \begin{Bmatrix} \gamma_1\mathbf{I} \\ \vdots \\ \gamma_n\mathbf{I} \end{Bmatrix} \mathbf{v}_0 \tag{4.59a}$$

$$\begin{Bmatrix} \mathbf{a}_1^{(r+1)} \\ \vdots \\ \mathbf{a}_n^{(r+1)} \end{Bmatrix} = \begin{bmatrix} \bar{\alpha}_{11}\mathbf{I} \cdots \bar{\alpha}_{1n}\mathbf{I} \\ \vdots \quad \vdots \quad \vdots \\ \bar{\alpha}_{n1}\mathbf{I} \cdots \bar{\alpha}_{nn}\mathbf{I} \end{bmatrix} \begin{Bmatrix} \mathbf{u}_1^{(r+1)} \\ \vdots \\ \mathbf{u}_n^{(r+1)} \end{Bmatrix} + \begin{bmatrix} \bar{\beta}_1\mathbf{I} \\ \vdots \\ \bar{\beta}_n\mathbf{I} \end{bmatrix} \mathbf{u}_0 + \begin{bmatrix} (\bar{\gamma}_1 + \beta_1)\mathbf{I} \\ \vdots \\ (\bar{\gamma}_n + \beta_n)\mathbf{I} \end{bmatrix} \mathbf{v}_0 + \begin{bmatrix} \gamma_1\mathbf{I} \\ \vdots \\ \gamma_n\mathbf{I} \end{bmatrix} \mathbf{a}_0 \tag{4.59b}$$

We note that solutions at the beginning of the iteration (i.e., $r = 0$) can be guessed by using the converged solutions of the previous time step to reduce the number of iterations. Proper guess of solutions can increase the efficiency of both iterative methods mentioned. In our study we use

$$\mathbf{u}_i^{(0)} = \mathbf{u}_0 + \tau_i \, \Delta t \mathbf{v}_0, \quad \text{for} \quad i = 1, 2, ..., n \tag{4.60a}$$

$$\mathbf{v}_i^{(0)} = \mathbf{v}_0 + \tau_i \, \Delta t \mathbf{a}_0, \quad \text{for} \quad i = 1, 2, ..., n \tag{4.60b}$$

$$\mathbf{a}_i^{(0)} = \mathbf{a}_0, \quad \text{for} \quad i = 1, 2, ..., n \tag{4.60c}$$

where $\mathbf{u}_0 = \mathbf{u}(t_s)$, $\mathbf{v}_0 = \dot{\mathbf{u}}(t_s)$, and $\mathbf{a}_0 = \ddot{\mathbf{u}}(t_s)$.

4.3 Analysis

4.3.1 Stability and Algorithmic Dissipation Control

We can use the single-degree-of-freedom problem given in Eq. (4.14) to check accuracies and stabilities of time integration algorithms. The algorithmic amplifica-

tion matrix ($^{a}\mathbf{A}$) can be obtained by applying a chosen time integration algorithm to the single-degree-of-freedom problem with $\xi = 0$ and $f(t) = 0$ (i.e., the undamped and unforced free vibrating case), and the spectral radius ($\rho(^{a}\mathbf{A})$) is defined as the maximum absolute value of the two eigenvalues of $^{a}\mathbf{A}$. Then the stability of the time integration algorithm can be investigated by checking the spectral radius for varying values of $\frac{\Delta t}{T}$, T being the period of the single-degree-of-freedom problem. A time integration algorithm is said to be unconditionally stable if $0 \leq \rho(^{a}\mathbf{A}) \leq 1$ is satisfied for all values of Δt. Figure 4.6 shows that the current algorithms are unconditionally stable if μ is chosen in the range of $0 \leq \mu \leq 1$.

As presented in Figure 4.6, the algorithmic dissipation level in the high frequency limit can be adjusted through the specification of μ. Algorithmic dissipations can be used to obtain stable solutions in highly nonlinear problems and to filter out the spurious high frequency responds caused due to the inaccurate spatial discretizations of original governing PDEs.

4.3.2 Accuracy

We already imposed desired order of accuracies on the new algorithms through the optimization of the weight parameters by comparing the entries of the exact and algorithmic amplification matrices. In fact, the accuracy condition given in Eq. (4.23) is valid for the numerical solutions of the first step, because it was defined by comparing the exact and algorithmic amplification matrices defined to relate the discrete solutions and the initial conditions. To define the order of accuracy of algorithms at any arbitrary time steps, however, a proper measure of the local error should be defined. In this case, the local truncation error considered in Refs. [5] can be used, because it defines the local error by using three adjacent discrete solutions

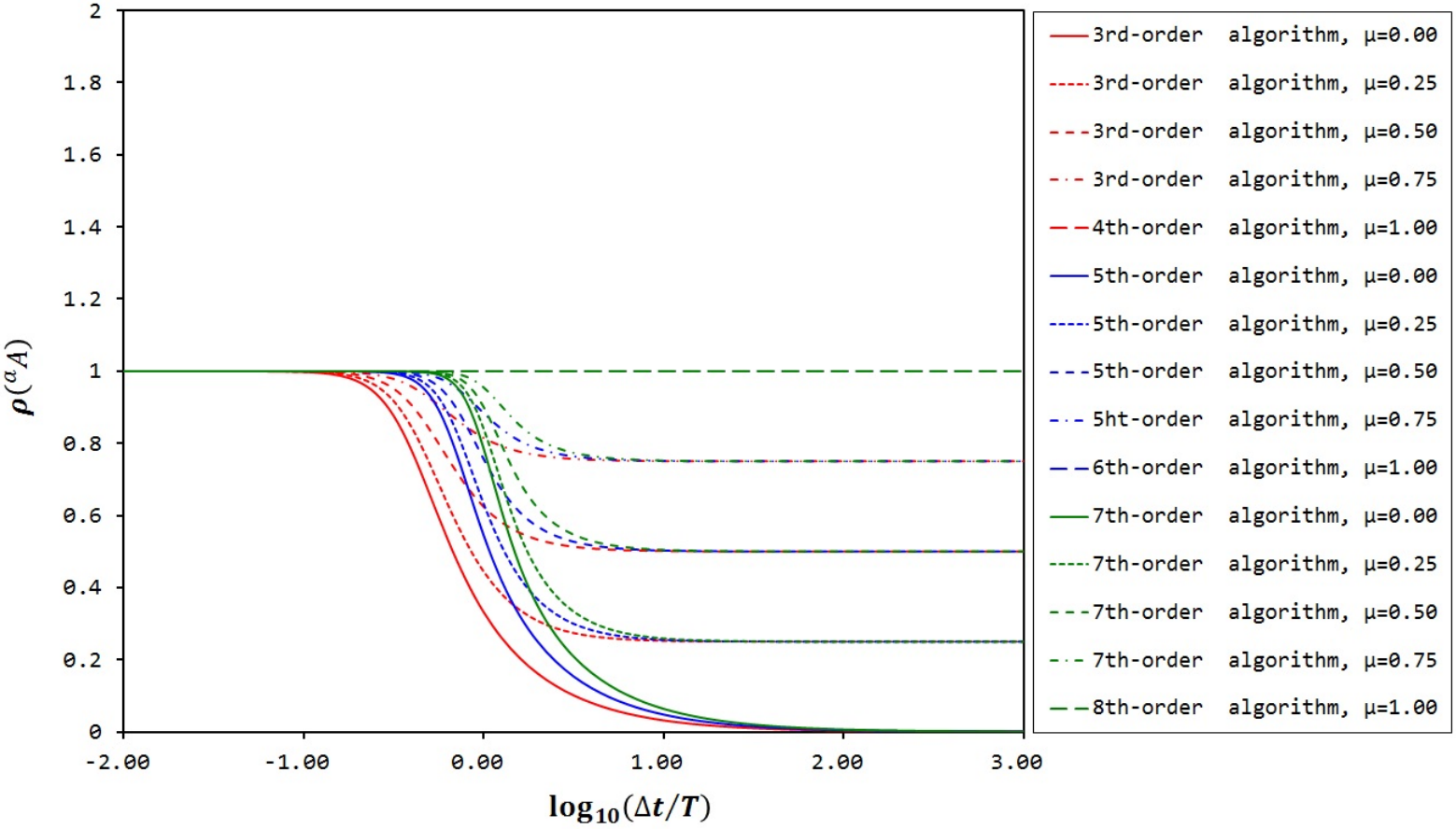

Figure 4.6: Spectral radii versus $\Delta t/T$ for current 3rd-, 5th-, and 7th-order algorithms with varying values of μ.

at any arbitrary steps. The local truncation error of the algorithm can be defined as

$$\tau_e(t_s) = \frac{1}{\Delta t^2}\Big(u(t_s + \Delta t) - 2A_1 u(t_s) + A_2 u(t_s - \Delta t)\Big) \tag{4.61}$$

where $A_1 = \frac{1}{2}\,\mathrm{tr}({}^a\mathbf{A})$, and $A_2 = \det({}^a\mathbf{A})$, and $u(t)$ is the exact solution of the single-degree-of-problem given in Eq. (4.16) for the unforced case (i.e., the case of $f(t) = 0$). Here, $\mathrm{tr}({}^a\mathbf{A})$ is the trace of ${}^a\mathbf{A}$, and $\det({}^a\mathbf{A})$ is the determinant of ${}^a\mathbf{A}$.

From Eq. (4.61), the order of accuracy of an algorithm is defined as kth-order, if $\tau_e = \mathrm{O}\left(\Delta t^k\right)$ is provided. Again, we note that the order of accuracy of the current algorithm obtained from the proposed procedure is $(2n-1)$th-order, if the nth-degree Lagrange interpolation functions are used for the approximations of time dependent variables. And $(2n)$ th-order accuracy can be obtained for the non-dissipative case (i.e., the case of $\mu = 1$).

4.4 Examples

4.4.1 Bi-Material Bar Problem

As a linear example, we consider the elastic bi-material bar problems whose cross section is unit square. The axial motion of the bar can be described by

$$\rho A\,\frac{\partial^2 u(x,t)}{\partial t^2} - \frac{\partial}{\partial x}\left(E(x)A\,\frac{\partial u(x,t)}{\partial x}\right) = f_0, \quad 0 \le x \le L, \quad t \ge 0 \tag{4.62}$$

with the initial and boundary conditions of

$$\begin{aligned} u(x,0) = 0, \quad \dot{u}(x,0) = 0 \\ u(0,t) = a\,\sin(\omega_p\, t), \quad u(L,t) = 0 \end{aligned} \tag{4.63}$$

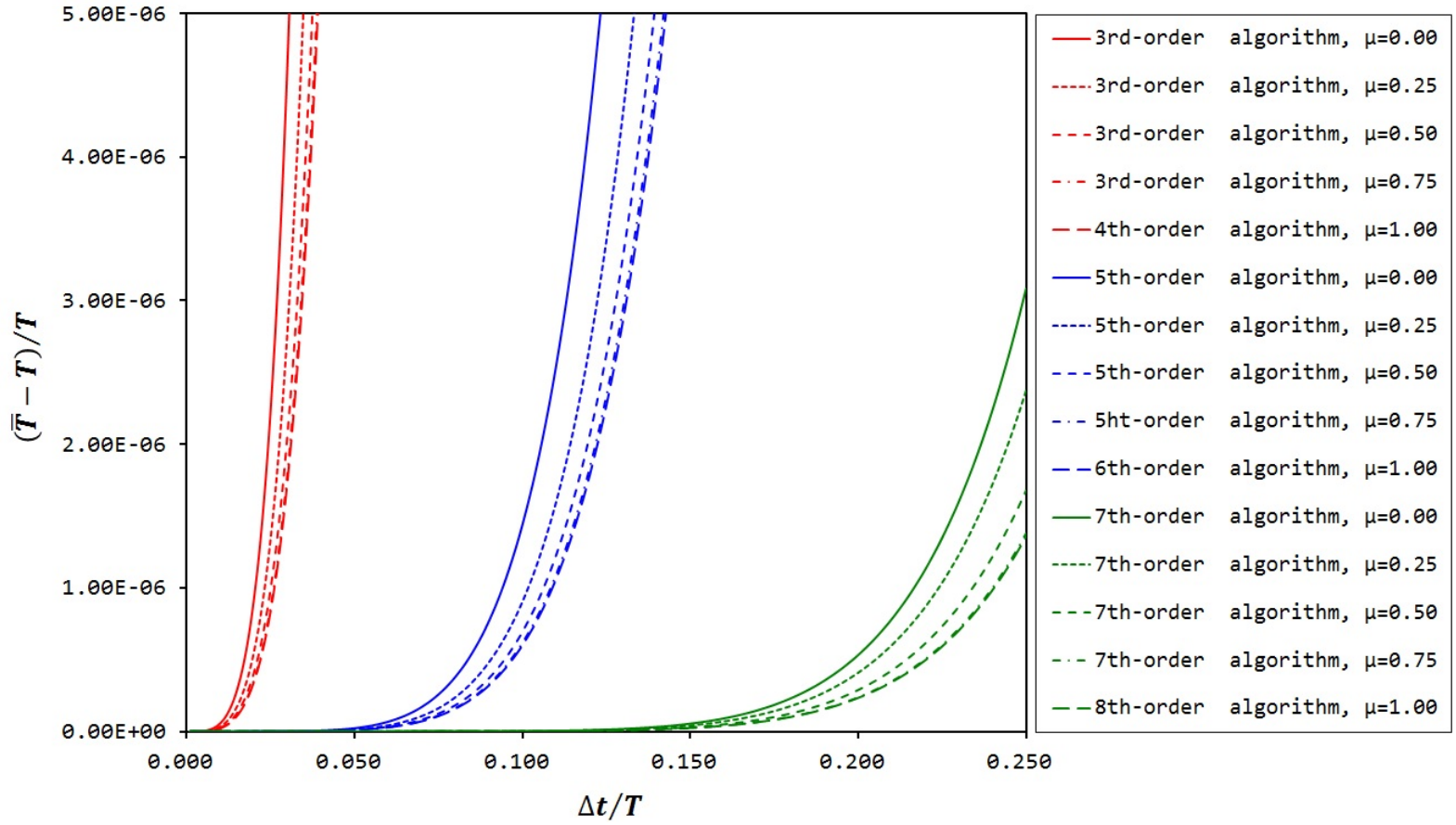

Figure 4.7: Relative period errors versus $\Delta t/T$ for current 3rd-, 5th-, and 7th-order algorithms with varying values of μ.

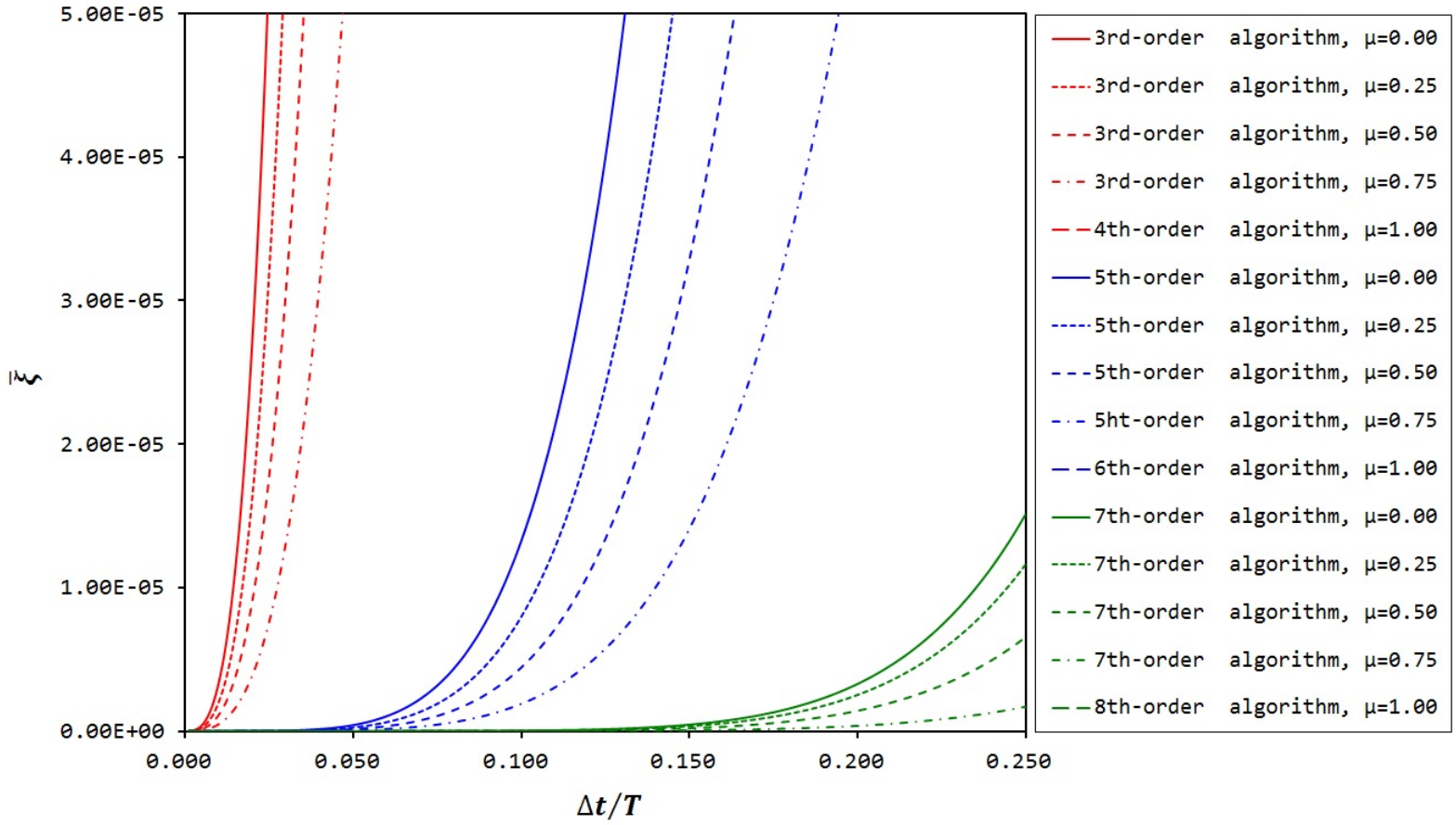

Figure 4.8: Damping ratios versus $\Delta t/T$ for current 3rd-, 5th-, and 7th-order algorithms with varying values of μ.

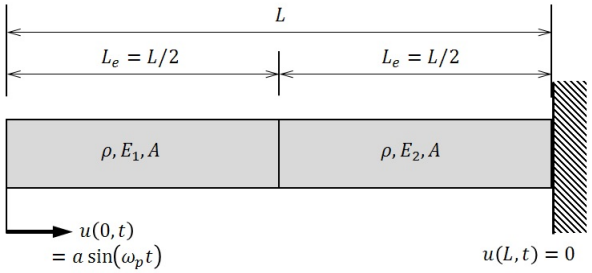

Figure 4.9: Description of bi-material bar with continuous excitation on left edge

where dimensionless constant properties of $\rho = 1$, $A = 1$, $L = 1$, $f_0 = 0$ are commonly used for both sides of the bar. $E(x) = E_1$ is used for the left half of the bar ($0 \leq x \leq L/2$), and $E(x) = E_2$ is used for the right half of the bar ($L/2 < x \leq L$). For the left edge boundary condition, $u(0,t) = a\ \sin(\omega_p\ t)$ are used with $a = 1/10$ and $\omega_p = 2\ \pi$.

For the spatial discretization, the weak-form Galerkin method[58, 1] is employed. With the quadratic Lagrange approximation of the displacement, the element level matrices and force vector of the semi-discrete equation are given as

$$\begin{aligned} \mathbf{M}_e &= \frac{\rho A L_e}{30}\begin{bmatrix} 4 & 2 & -1 \\ 2 & 16 & 2 \\ -1 & 2 & 4 \end{bmatrix}, \quad \mathbf{C}_e = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \mathbf{K}_e &= \frac{E_e A}{3L_e}\begin{bmatrix} 7 & -8 & 1 \\ -8 & 16 & -8 \\ 1 & -8 & 7 \end{bmatrix}, \quad \mathbf{f}_e = \frac{f_0\ L_e}{6}\begin{Bmatrix} 1 \\ 4 \\ 1 \end{Bmatrix} \end{aligned} \tag{4.64}$$

In current case, we use two equal size quadratic elements for spatial discretization.

Then the assembled global semi-discrete equation is obtained as

$$\begin{aligned}
&\frac{1}{60}\begin{bmatrix} 4 & 2 & -1 & 0 & 0 \\ 2 & 16 & 2 & 0 & 0 \\ -1 & 2 & 8 & 2 & -1 \\ 0 & 0 & 2 & 16 & 2 \\ 0 & 0 & -1 & 2 & 4 \end{bmatrix}\begin{Bmatrix} \ddot{u}_1 \\ \ddot{u}_2 \\ \ddot{u}_3 \\ \ddot{u}_4 \\ \ddot{u}_5 \end{Bmatrix} \\
&+\frac{2}{3}\begin{bmatrix} 7\,E_1 & -8\,E_1 & E_1 & 0 & 0 \\ -8\,E_1 & 16\,E_1 & -8\,E_1 & 0 & 0 \\ E_1 & -8\,E_1 & 7\,(E_1+E_2) & -8\,E_2 & E_2 \\ 0 & 0 & -8\,E_2 & 16\,E_2 & -8\,E_2 \\ 0 & 0 & E_2 & -8\,E_2 & 7\,E_2 \end{bmatrix}\begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix}
\end{aligned} \tag{4.65}$$

By imposing $u_1 = 1/10\,\sin(2\,\pi\,t)$, $\ddot{u}_1 = -4/10\,\pi^2\,\sin(2\,\pi\,t)$, $u_5 = 0$, and $\ddot{u}_5 = 0$ on Eq. (4.65), we can reduce Eq. (4.65) to

$$\begin{aligned}
&\frac{1}{30}\begin{bmatrix} 8 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 8 \end{bmatrix}\begin{Bmatrix} \ddot{u}_2 \\ \ddot{u}_3 \\ \ddot{u}_4 \end{Bmatrix} + \frac{2}{3}\begin{bmatrix} 16\,E_1 & -8\,E_1 & 0 \\ -8\,E_1 & 7\,(E_1+E_2) & -8\,E_2 \\ 0 & -8\,E_2 & 16\,E_2 \end{bmatrix}\begin{Bmatrix} u_2 \\ u_3 \\ u_4 \end{Bmatrix} \\
&= \frac{1}{150}\begin{Bmatrix} 80\,E_1\,\sin(2\,\pi\,t) + 2\,\pi^2\,\sin(2\,\pi\,t) \\ -10\,E_1\,\sin(2\,\pi\,t) - \pi^2\,\sin(2\,\pi\,t) \\ 0 \end{Bmatrix}
\end{aligned} \tag{4.66}$$

Now, Eq. (4.66) can be used to test new algorithms for varying values of E_1 and E_2.

Two important performances of new algorithms can be tested by using this linear problem given in Eq. (4.66). First, we can test the long-term performance of algorithms. By setting $E_1 = 10$ and $E_2 = 1$, we can synthesize a bi-material bar

vibration problem that has moderate differences in natural frequencies. It is not that difficult to find the modal solution of Eq. (4.66), and the modal solution can be used as a reference solution. Since the main purpose of this problem is to test the performance of new time integration algorithms, the modal solutions of Eq. (4.66) is used as the reference solution, instead of the analytical solution of the original PDE given in Eq. (4.62).

For the first case, we choose a considerably small size of time step for the trapezoidal rule and a considerably large size of time step for new algorithms. Usually, the proper size of Δt can be determined by using the maximum natural frequency of the semi-discrete system. For the trapezoidal rule, we use one tenth of the period associated with the maximum natural frequency. Since the maximum natural frequency of the first case is 29.0581, the corresponding period is computed as 0.0344138. Then the time step of the trapezoidal rule can be selected as $\Delta t = 0.00344138$. The time step of new 10th-order algorithm is selected as $\Delta t = 0.0860345$ (25 times of the time step of the trapezoidal rule) by considering sizes of the computer memories required by the trapezoidal rule and the 10th-order algorithm. These typical choices of time steps can equalize overall computer memories required in each of the methods. To be specific, the current 10th-order algorithm requires 25 times more memory to store the effective stiffness compared to the trapezoidal rule, because a $5m \times 5m$ effective coefficient matrix should be used in the 10th-order algorithm while a $m \times m$ effective coefficient matrix can be used in the trapezoidal rule. The center displacement (u_3), velocity ($\dot{u}_3$) and acceleration ($\ddot{u}_3$) at $t = 100$ (after hundred cycles of excitation at the left edge) are presented in Figs. 4.13-4.15.

As shown in Figs. 4.10 -4.12, two numerical solutions presented almost identical results for the first period. However, after hundred cycles of excitation, the numerical solution (especially, the velocity and acceleration solutions) of the trapezoidal rule at

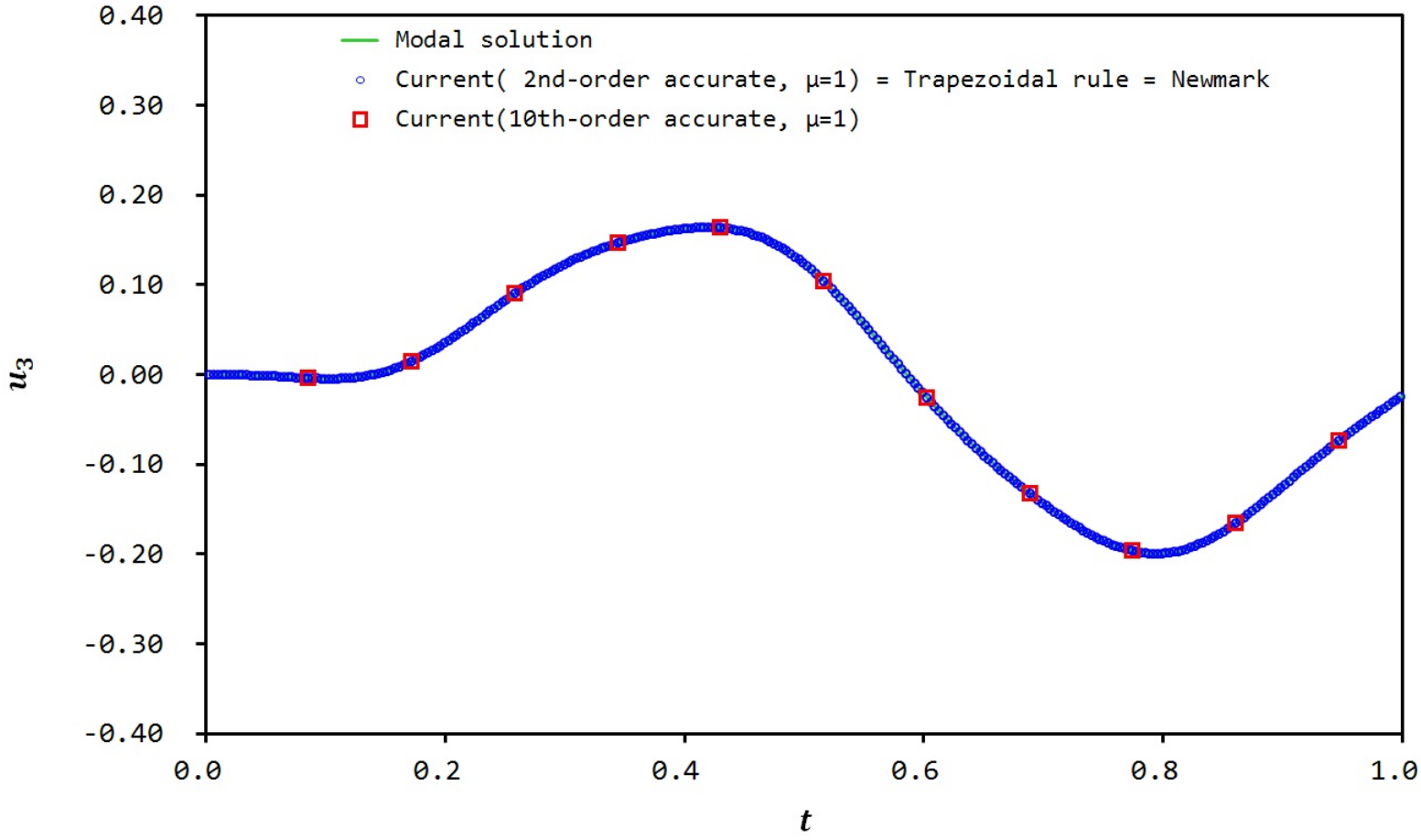

Figure 4.10: Comparison of center displacements at the beginning of excitation. $(0 \leq t \leq 1)$

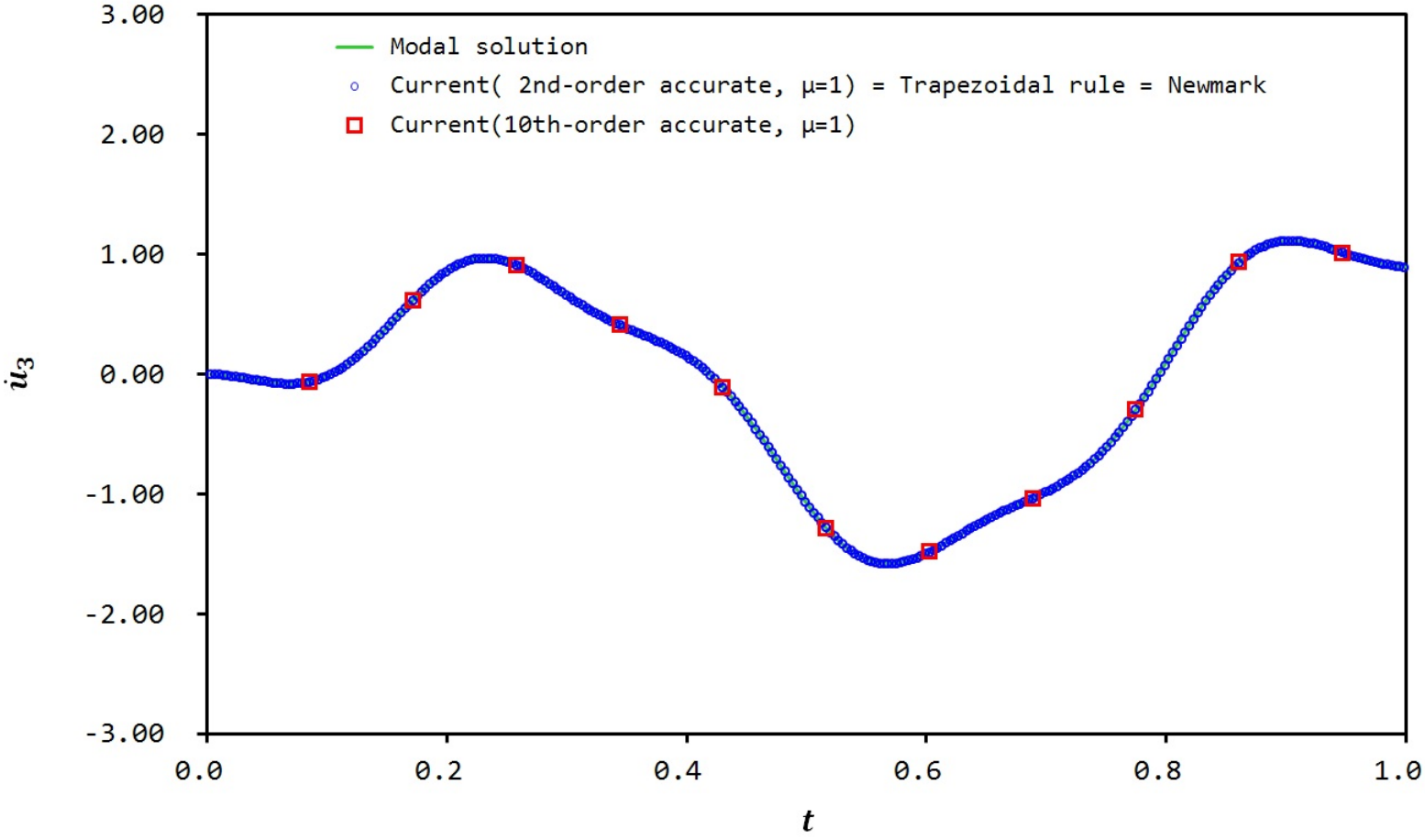

Figure 4.11: Comparison of center velocities at the beginning of excitation. $(0 \leq t \leq 1)$

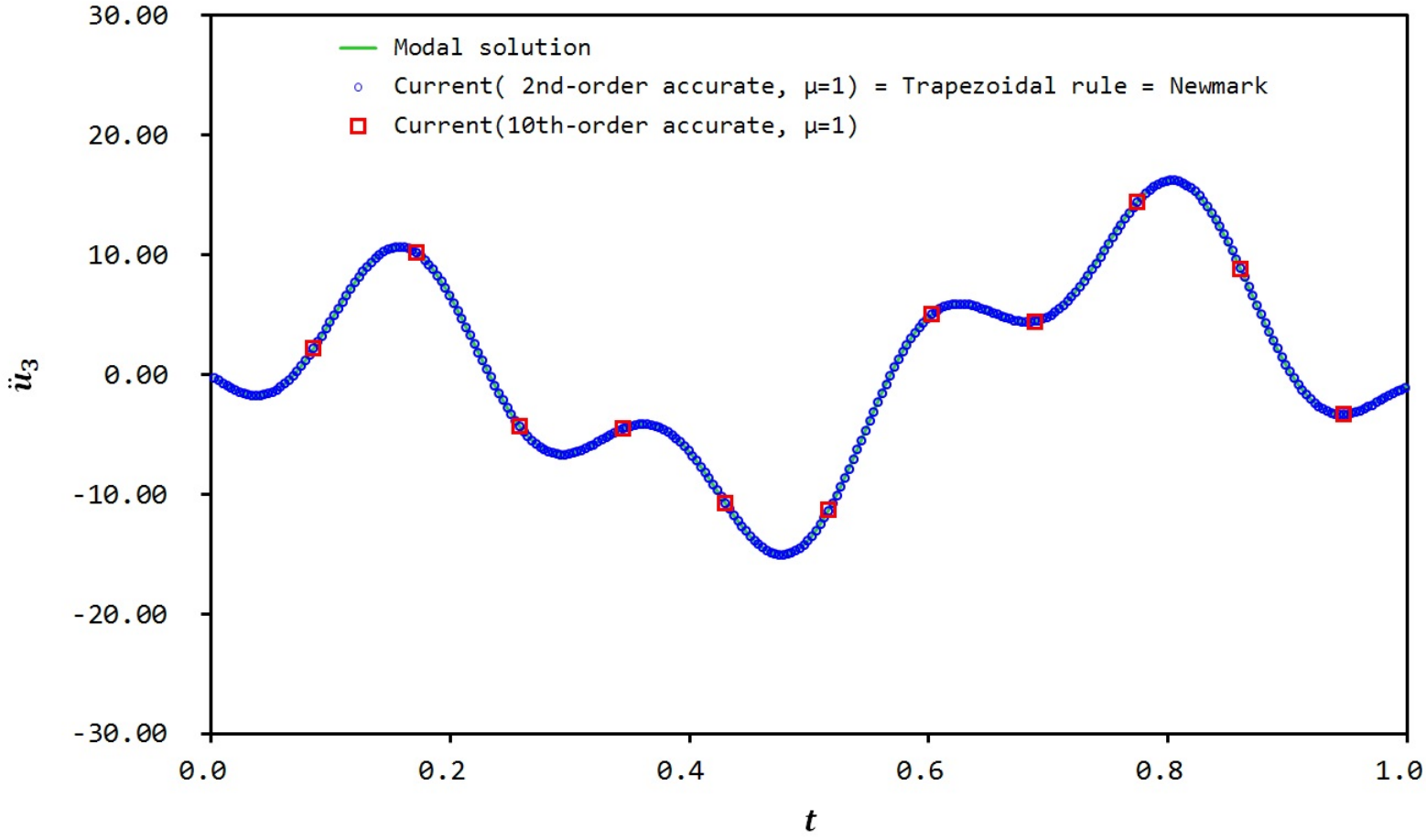

Figure 4.12: Comparison of center accelerations at the beginning of excitation. ($0 \leq t \leq 1$)

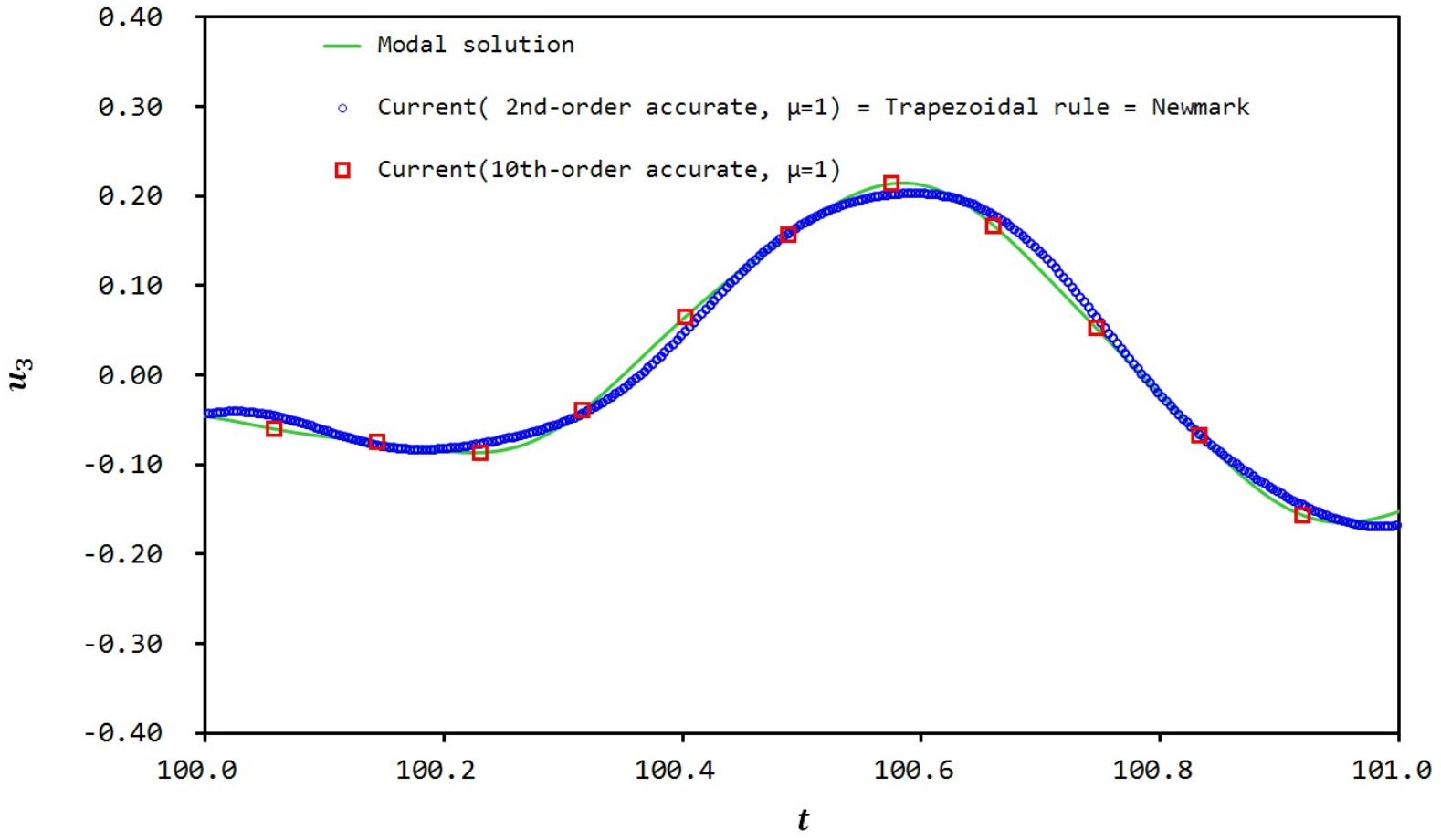

Figure 4.13: Comparison of center displacements at $t = 100.0$ (after hundred cycles of excitation).

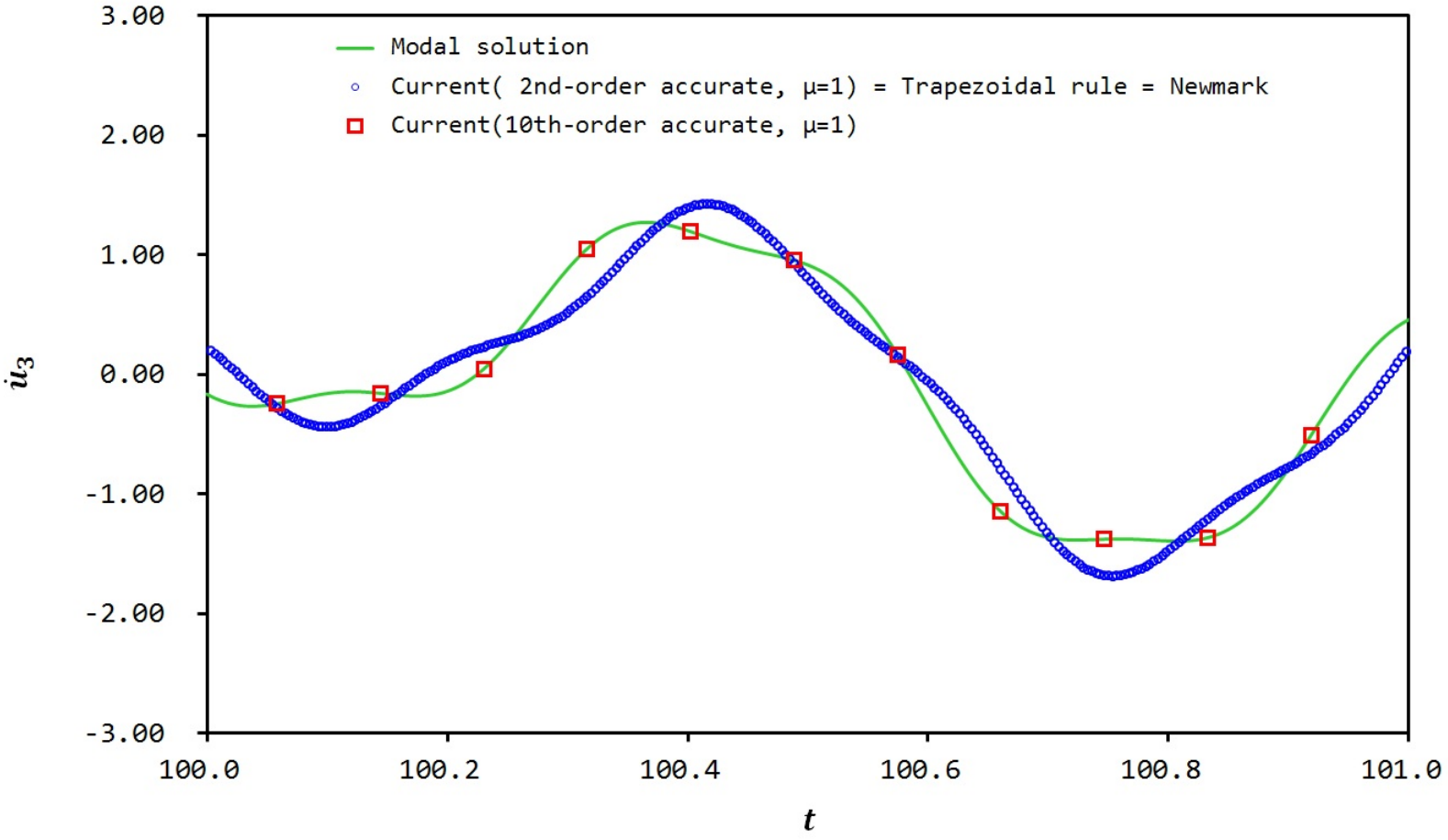

Figure 4.14: Comparison of center velocities at $t = 100.0$ (after hundred cycles of excitation).

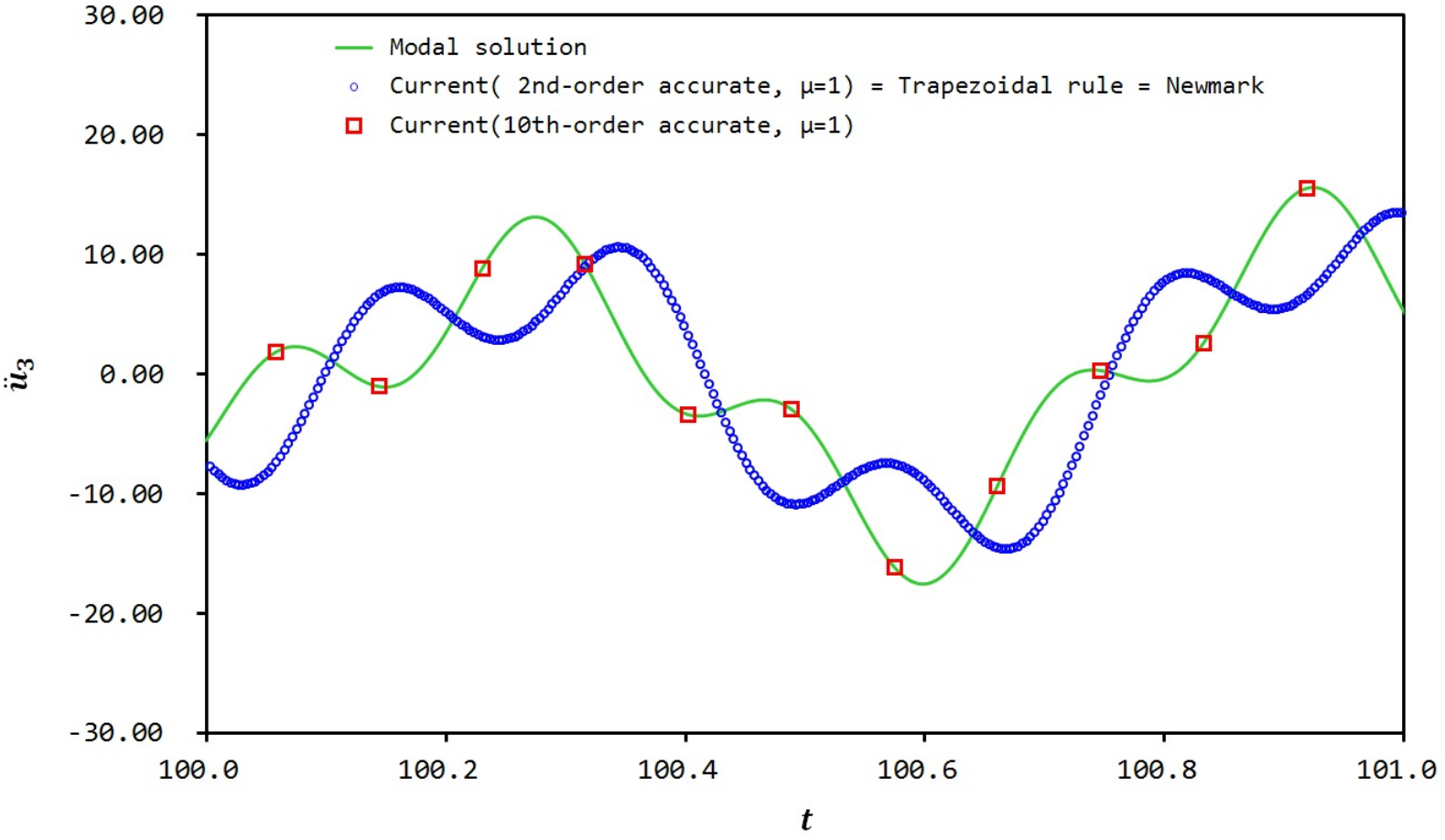

Figure 4.15: Comparison of center accelerations at $t = 100.0$ (after hundred cycles of excitation).

the center of the bar presented noticeable errors compared with the modal solution, while the numerical solution of the 10th-order algorithm perfectly superposed the modal solution as presented in Figs. 4.13 -4.15. Thus, in this long term analysis, higher-order algorithms can be more advantageous allowing use of much larger time steps. From Eqs. (4.41) -(4.42), α_{ij}, β_i and γ_i of current algorithms can be computed.

For the second case, the high-frequency filtering capability of algorithms is tested by setting the left half of the bar very stiff. As stated previously, poor spatial discretization of original PDEs can produce spurious high frequency modes in numerical solutions. To reproduce a similar situation (i.e., spurious high frequency mode) in our test problem, we intensionally make the left half of the bar very stiff. In our numerical experiment, the highest and second highest frequency modes of Eq. (4.66) are assumed to be the spurious high frequency modes. Similar types of numerical experiments have been conducted by using discrete mass-spring system problems by Hilber and Hughes [6].

In fact, high frequency filtering can also be done through post-processing (called low pass filtering [22]) of numerical solutions, but the use of algorithmic dissipations has been considered a more practical and safer way than post-processing of numerical solutions in the literature [36]. In time integration algorithms, the spurious high frequency effect can be filtered out by utilizing controllable algorithmic dissipation with proper choice of time steps. To get the maximum dissipation in the high frequency range, we set $\mu = 0$ for the current algorithms.

In the second problems, we use $E_1 = 5,000$, and all other properties are kept the same as the first case. To check the high frequency filtering capability, two important capabilities of algorithms should be observed in their numerical solutions. Most of all, algorithms should be able to effectively eliminate the assumed spurious high frequency effect, and at the same time they also should be able to present the important

low frequency mode accurately. Especially, by eliminating all high frequency effects, we wish to observe that u_3 moves along with the prescribed displacement at the left edge (i.e., $u_1 = 1/10 \sin(2\ \pi\ t)$) as if the left half of the bar is "rigid".

Three natural frequencies obtained from the modal decomposition analysis of the fully discrete system given in Eq. (4.66) are 638.295, 198.231, and 6.32324. The frequency of prescribed excitation at the left edge is 6.28319. We wish to eliminate the effects associated with the high frequencies 638.295 and 198.231 from numerical solutions during the time integration, while preserving the effects associated with the low frequencies 6.32324 and 6.28319 in numerical solutions. For the elimination of the high frequency effect, we chose Δt as 0.156667 ($= \frac{1}{638.295} \times 100 = \frac{1}{198.231} \times 31.056$) introducing enough algorithmic damping into the two high frequency modes. The Baig and Bathe method [35, 8] is also used as the second order algorithm case for the comparison of numerical solutions. The Baig and Bathe method is known to be one of the most effective second-order algorithms, which can be used for the spurious high frequency filtering.

It can be observed that the displacement solutions (u_3) are almost superposing each other as presented in Fig. 4.16, however, the velocity and acceleration solutions (v_3 and a_3) are showing noticeable differences between algorithms as presented in Figs. 4.17 -4.19. Especially the acceleration solution obtained from the Baig and Bathe method presented the largest period error when it is compared with the reference solution. To fix this, we reduced the size of time step to $\Delta t/10$ in the Baig and Bathe method, then the high frequency effects were not filtered effectively as shown in Figs. 4.17 -4.19. Thus, in this particular case, it can be said that the chosen second-order algorithm cannot give a reasonably good numerical solution by adjusting the time step. On the other hand, the 7th- and 9th-order algorithms presented reasonably good numerical solutions eliminating high frequency effects very

effectively as show in Figs. 4.17 -4.19.

4.4.2 Simple Pendulum

As a nonlinear example, the nonlinear oscillation of the simple pendulum [44] is considered. The motion of the simple pendulum can be described as

$$\ddot{\theta} + \omega^2 \sin(\theta) = 0 \tag{4.67}$$

with the initial angle and the initial angular velocity

$$\theta(0) = \theta_0 \tag{4.68a}$$

$$\dot{\theta}(0) = \dot{\theta}_0 \tag{4.68b}$$

where, $\omega = \sqrt{g/L}$, and $\theta(t)$ is the angle between the rod and a vertical line at t. g is the gravitational constant, and L is the length of the massless rod. In the current study we assumed a dimensionless case with $\omega = 1$.

This simple pendulum problem is very useful for the test of nonlinear performance of newly developed time integration algorithms, because important information pertaining to the problem (such as nonlinear period and maximum angle) can be exactly obtained with the initial conditions given in Eqs. (4.68a) and (4.68b). In addition to this, the degree of nonlinearity can also be adjusted by simply adjusting the initial conditions. Nonlinearity of the problem will become higher if large values of θ_0 and $\dot{\theta}_0$ are used. In the work of Fung [44, 70], some useful data of this problem have been exactly computed by manipulating the total energy of the pendulum in motion. Here, we briefly review useful discussions provided in the work of Fung[44]. For the pendulum in motion, the total energy (i.e., sum of potential and kinetic energies)

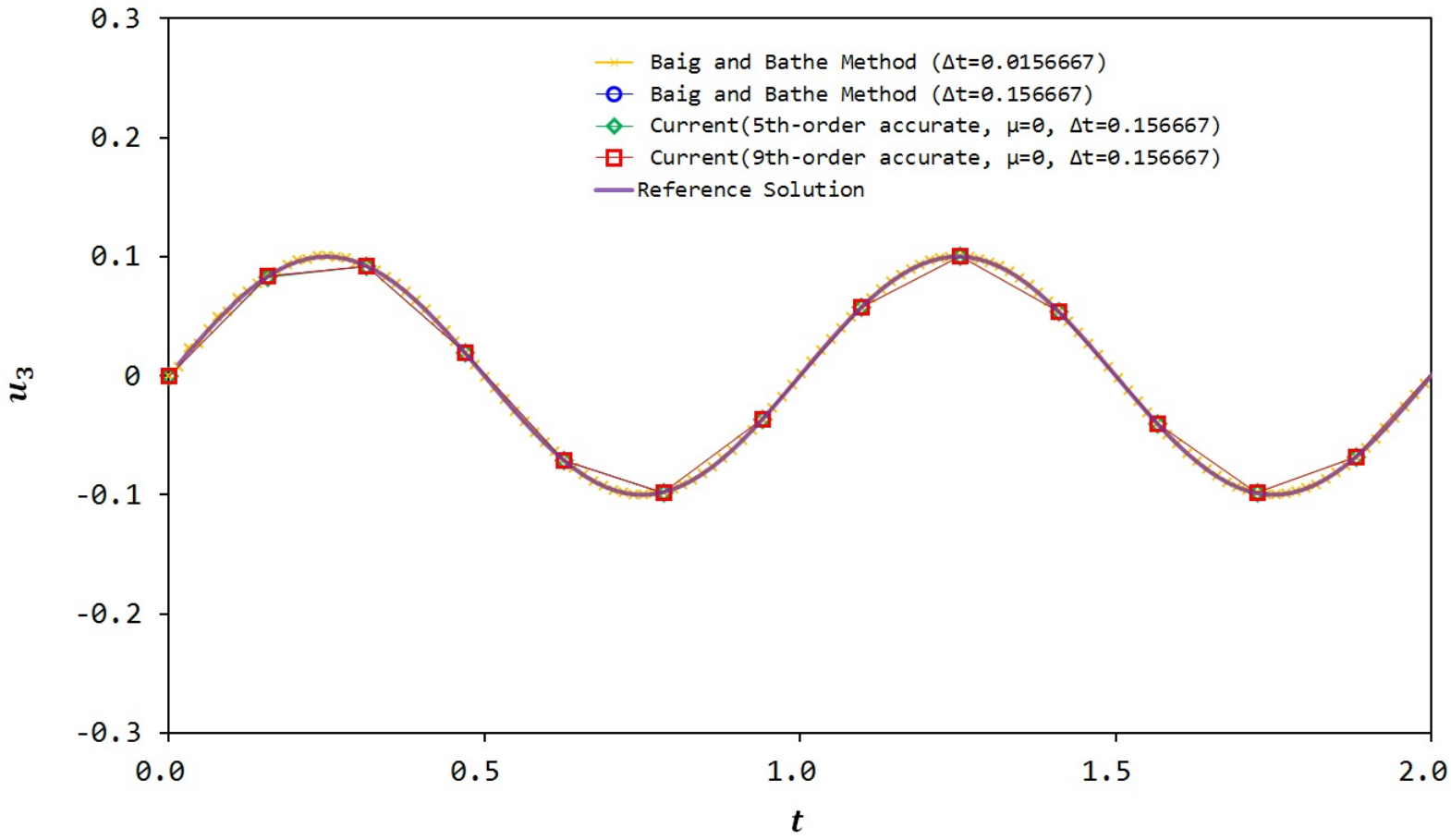

Figure 4.16: Comparison of center displacements of the bi-material bar with stiff and soft parts.

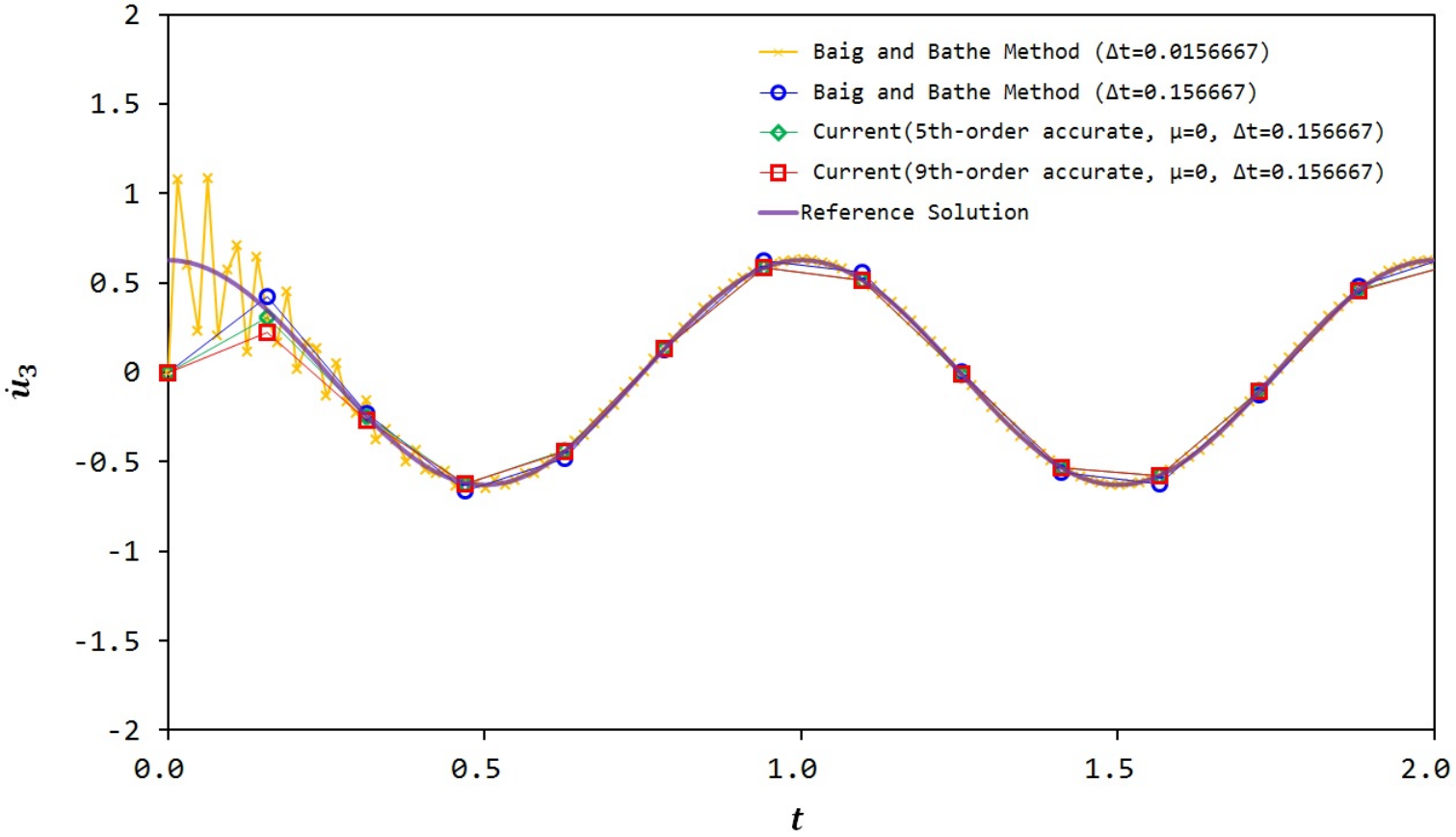

Figure 4.17: Comparison of center velocities of the bi-material bar with stiff and soft parts.

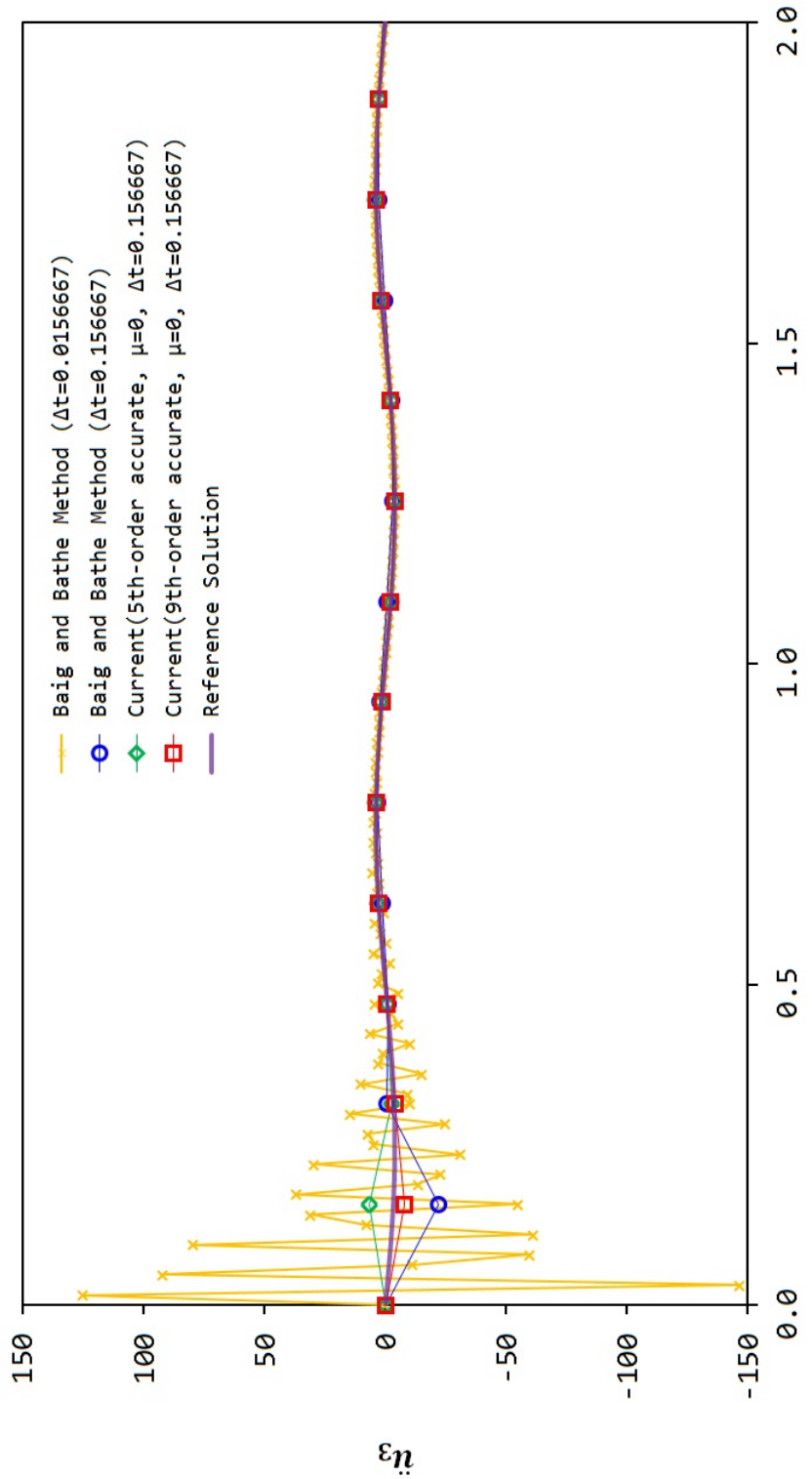

Figure 4.18: Comparison of center accelerations of the bi-material bar with stiff and soft parts.

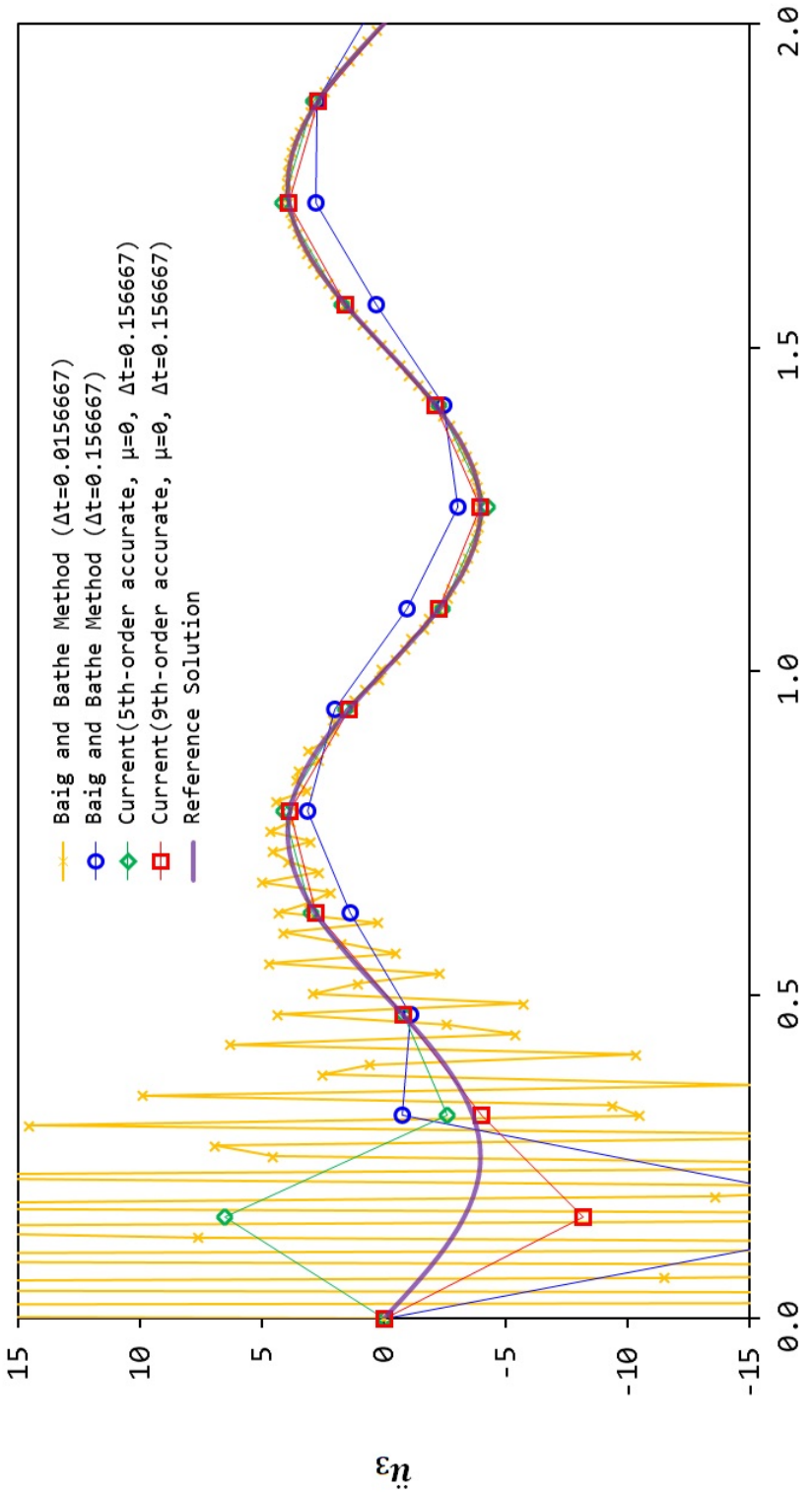

Figure 4.19: Enlarged picture of Fig. 4.18.

should always be conserved. The total energy can be expressed as

$$\frac{1}{2}\left(\frac{\mathrm{d}\theta}{\mathrm{d}t}\right)^2 - \omega^2\cos(\theta) = \text{a constant} = \frac{1}{2}\dot{\theta}_0^2 - \omega^2\cos(\theta_0) \tag{4.69}$$

Then, the relation of θ and t can be obtained as

$$t = \pm\int_{\theta_0}^{\theta} \frac{1}{\sqrt{\dot{\theta}_0^2 + 2\,\omega^2\left(\cos(\vartheta) - \cos(\theta_0)\right)}}\mathrm{d}\vartheta \tag{4.70}$$

where $\vartheta = \theta - \theta_0$. By assuming zero initial angle ($\theta_0 = 0$), Eq. (4.70) can be simplified as

$$t = \pm\frac{1}{\left|\dot{\theta}_0\right|}\int_0^{\theta} \frac{1}{\sqrt{1-\kappa^2\sin^2(\frac{\vartheta}{2})}}\mathrm{d}\vartheta = \frac{2}{\left|\dot{\theta}_0\right|}\mathrm{Ei}\left(\sin\left(\frac{\vartheta}{2}\right),\ \kappa\right) \tag{4.71}$$

where, $\kappa = 2\omega/\dot{\theta}_0$ and $\mathrm{Ei}\left(z,\ \kappa\right)$ is the elliptical integral of the first kind. If we assume that the pendulum keeps oscillating in the plane instead of rotating, the maximum angle $\theta_{\max}$ can be computed by setting $\frac{\mathrm{d}\theta}{\mathrm{d}t} = 0$ in Eq. (4.69). For the oscillating pendulum, $\theta_{\max}$ is computed as

$$\theta_{\max} = 2\,\sin^{-1}\left(\frac{\dot{\theta}_0}{2\,\omega}\right) \tag{4.72}$$

In our case, $\theta_{\max}$ is determined by the given initial velocity. The exact solution of the first quarter of the nonlinear period (i.e., $T_f = T/4$) can also be computed from Eq. (4.71) by finding values of t for varying values of θ. It should be also noted that T_f can be directly computed by substituting Eq. (4.72) into Eq. (4.71).

Two cases are considered for the test. First, the initial velocity has been chosen as $\dot{\theta}_0 = \sqrt{2}$ to get the maximum angle $\theta_{\max} = 90^\circ$ as shown in

Fig. 4.20.

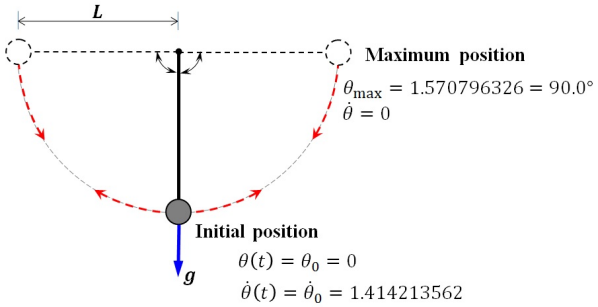

Figure 4.20: Oscillation of moderately nonlinear simple pendulum with $\theta_{\max} = 90°$.

Second, the initial velocity has been chosen as $\dot{\theta}_0 = 1.9999992384$ to get the maximum angle $\theta_{\max} = 179.9°$ as shown in Fig. 4.21.

The first case was considered to verify the performance of the current algorithms and well-known second-order algorithms for a moderate nonlinear case. The numerical solutions obtained from the current 8th- and 10th-order algorithms and two existing second-order algorithms are compared with the exact solution. For this moderate nonlinear case, all numerical solutions presented acceptable accuracy as shown in Fig. 4.22.

The second case was specially intended to demonstrate advantages of using current higher-order algorithms in highly nonlinear cases. Since we set the initial velocity to get the maximum angle of 179.9°, energy conservation is very critical for the pendulum to continue oscillation in the plane. The pendulum may rotate around the pivot point, if the total energy of the system slightly increases due to errors caused

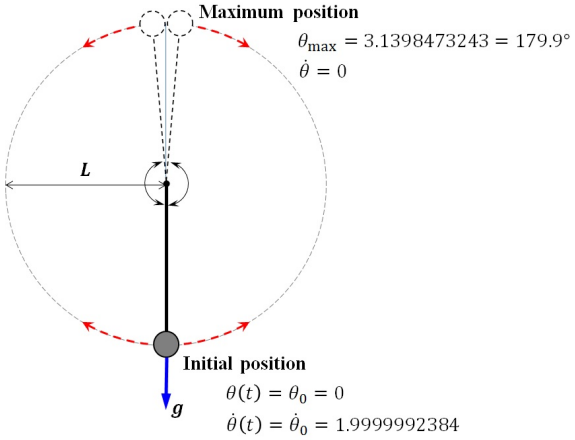

Figure 4.21: Oscillation of highly nonlinear simple pendulum with $\theta_{\max} = 279.9°$.

by time integration algorithms. This interesting case has been presented in Fig. 4.23.

First, we used $\Delta t = T/10,000$ for the chosen second-order algorithms. Even though very small time step has been used for second-order algorithms, the pendulum failed to oscillate in the plane in the second-order algorithms tested. However, the new 8th- and 10th-order algorithms gave very accurate solutions even with considerably large time steps ($\Delta t = T/100$).

If a reduced size of time step (i.e., $\Delta t = T/20,000$) is used for second-order algorithms, the pendulum did not rotate, but the period errors became noticeable as presented in Fig. 4.24. For more sophisticated comparison, numerical solutions obtained from various methods have been presented in Table 4.3 along with the exact solution.

In Table 4.3, it can be observed that new algorithms can provide much better accuracy than second-order algorithms for the highly nonlinear case of the pendulum problem. The numerical result of the new algorithms also presented good agreement with the equivalent algorithms of Fung [44].

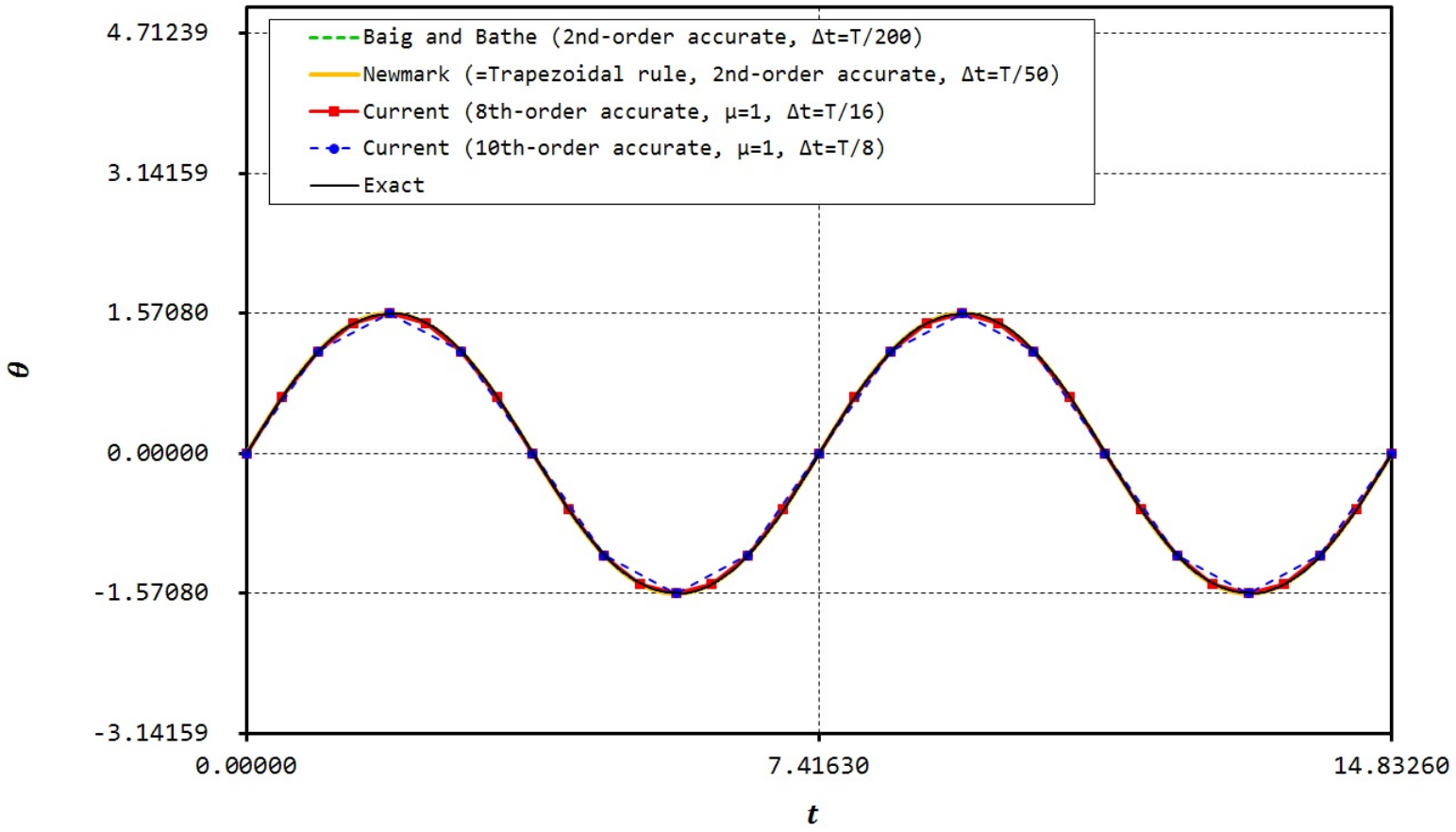

Figure 4.22: Comparison of angles for current (8th-, and 10th-order) and second-order algorithms (the Newmark method and the Baig and Bathe method) for mildly nonlinear case. Initial conditions are chosen to get the maximum angle $\pi/2 = 90.0^\circ$ at the quarter of period.

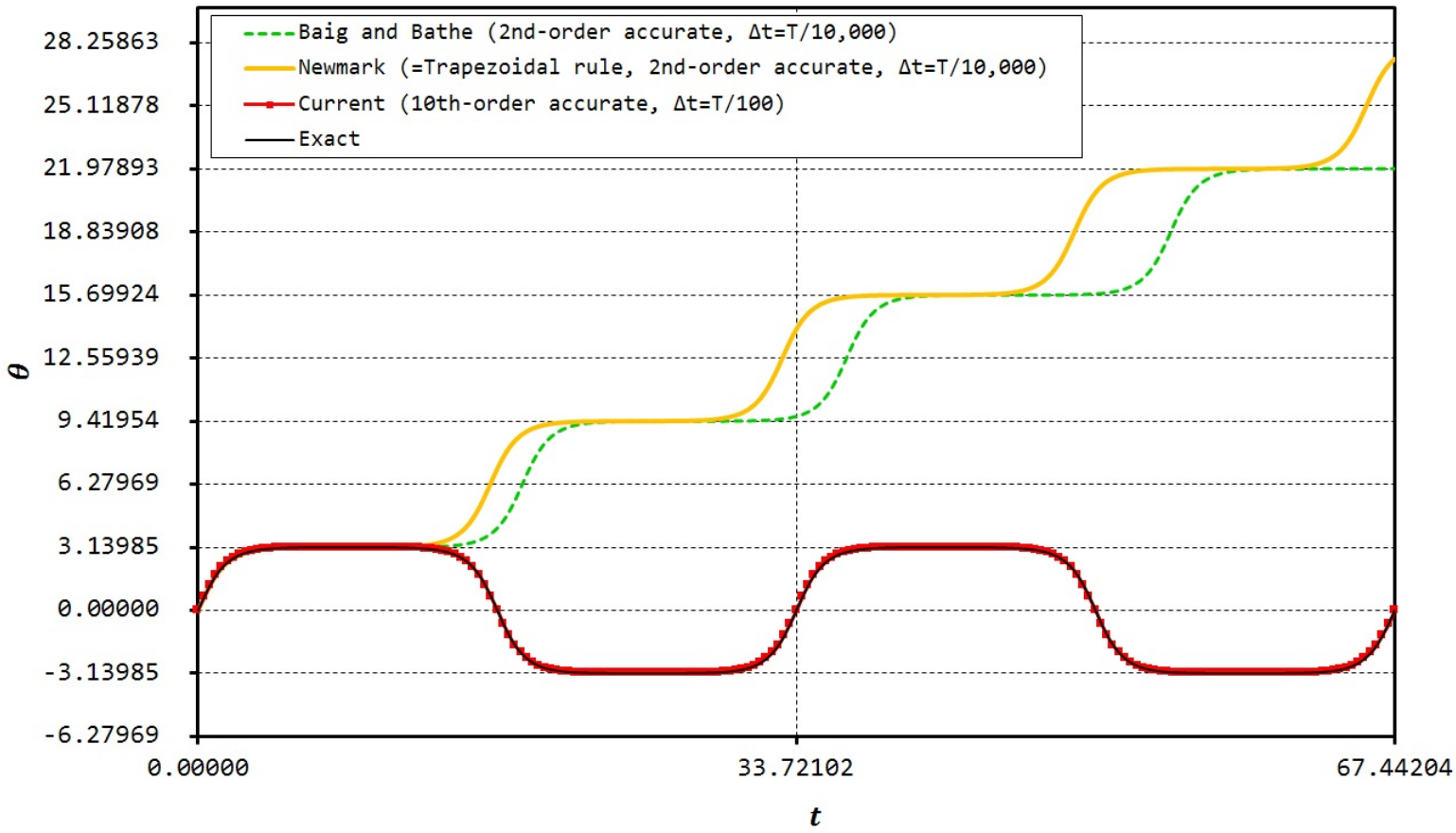

Figure 4.23: Comparison of angles for current (8th-, and 10th-order) and second-order algorithms (the Newmark method and the Baig and Bathe method) for highly nonlinear case. The second-order algorithms used $\Delta t = T/10,000$, and the current 10th-order algorithm used $\Delta t = T/100$, respectively, T being the period.

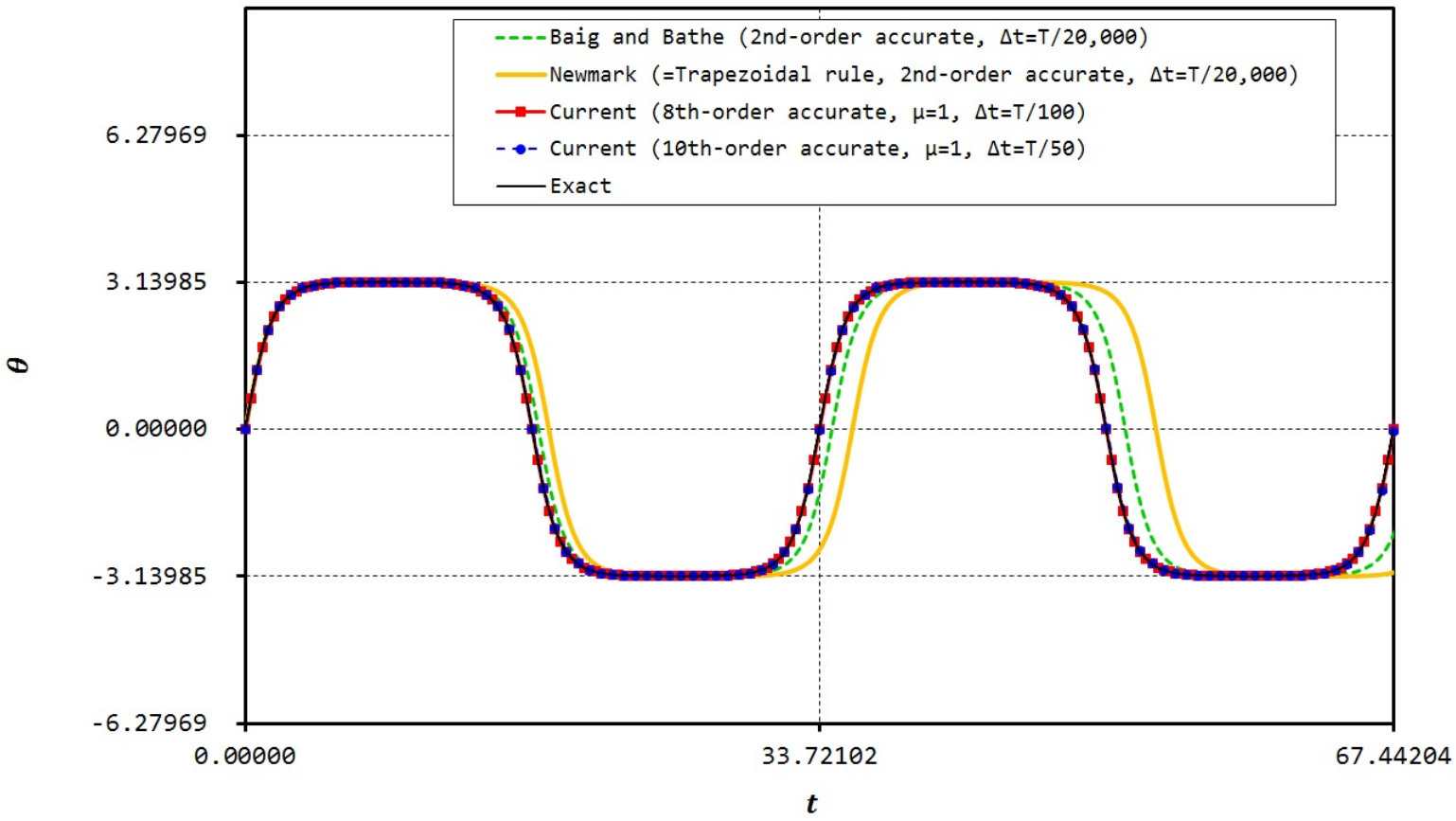

Figure 4.24: Comparison of angles for current (8th-, and 10th-order) and second-order algorithms (the Newmark method and the Baig and Bathe method) for highly nonlinear case. The second-order algorithms used $\Delta t = T/20,000$, and the current 8th- and 10th-order algorithms used $\Delta t = T/100$ and $\Delta t = T/50$, respectively, T being the period.

Method(order of accuracy)	$\frac{T_f}{\Delta t}$	$\theta(T_f)$	error = $\frac{\text{exact}-\theta(\text{T}_\text{f})}{\text{exact}}$
exact	-	3.139847324	-
current(5th, $\mu = 0$)	25	3.136116024	-0.118837e-2
	50	3.139724061	-0.392578e-4
current(6th, $\mu = 1$)	25	3.140101399	-0.809196e-4
	50	3.139851077	0.119518e-5
current(7th, $\mu = 0$)	10	3.132065933	0.247827e-2
	25	3.139833891	0.427840e-5
current(8th, $\mu = 1$)	10	3.139011835	0.266092e-3
	25	3.139846872	0.144169e-6
current(9th, $\mu = 0$)	5	3.103983662	-0.114221e-1
	10	3.139731676	-0.368326e-4
current(10th, $\mu = 1$)	5	3.128483768	-0.361914e-2
	10	3.139848406	0.344411e-6
Fung[44](5th, $\mu = 0$)	25	3.135946967	0.124221e-2
	50	3.139721358	0.401186e-4
Fung(6th, $\mu = 1$)	25	3.139656436	0.607953e-4
	50	3.139844510	0.896197e-6
Fung(7th, $\mu = 0$)	10	3.123150250	0.531779e-2
	25	3.139834046	0.422900e-5
Fung(8th, $\mu = 1$)	10	3.140510309	-0.211152e-3
	25	3.139847686	-0.115138e-6
Fung(9th, $\mu = 0$)	5	3.058777884	0.258195e-1
	10	3.140080122	0.741430e-4
Fung(10th, $\mu = 1$)	5	3.149078105	0.293988e-2
	10	3.139846547	-0.247731e-6
Newmark[23](2nd)	500	3.194151076	0.172950e-1
	1,000	3.153421369	0.432315e-2
	2,500	3.142019059	0.691669e-3
	5,000	3.140390264	0.172919e-3
Baig and Bathe[35](2nd)	500	3.166961328	0.863545e-2
	1,000	3.146629692	0.216009e-2
	2,500	3.140932907	0.345744e-3
	5,000	3.140118751	0.864459e-4

Table 4.3: Comparison of nonlinear numerical solutions at $t = T/4 = 8.430255141$ for various methods in highly nonlinear case ($\dot{\theta}_0 = 1.999999238$, $\theta_{\max} = 3.139847324$).

4.5 Conclusion

In this study, the unconventional mixed formulation and the modified weighted residual method have been considered to develop new time integration algorithms. Due to the unconventional setting of computational framework, newly developed algorithms can achieve desirable computational structures and provide many preferable attributes (such as unconditional stability, controllable algorithmic dissipation, improved accuracy, and easy implementation), at the same time, eliminating some shortcomings of the existing higher-order algorithms as discussed in the text.

Some improvements of new algorithms are:

(1) The new algorithms can be written down in the ready-to-use forms, which can be readily implemented into computer codes without any additional procedures or undetermined parameters.

(2) Due to the unique computational framework used, the new algorithms can also be easily applied to nonlinear problems without any modifications, which is clearly not possible in the existing algorithms developed based on the traditional weighted residual method.

(3) Due to the simple and intuitive computational structures of the new algorithms, other types of value problems (other than the second-order initial value problems) can also be tackled in the same unified manner.

To demonstrate advantages of using new higher-order algorithms, the simple linear and nonlinear numerical examples were considered. In some extreme situations, such as the long-term analysis and the fast filtering of the spurious high frequencies, the existing second-order algorithms could not provide reasonably good predictions, but use of the higher-order algorithms could. Especially, if the spurious high frequency mode is located relatively close to the important low frequency mode, then

the range of the admissible sizes of time steps becomes very limited. In this kind of situation, the second-order algorithms cannot provide good accuracy for the important low frequency mode and effective filtering of the spurious high frequency mode simultaneously, as shown in the second linear example.

As explained in the text, the new algorithms can be easily applied to nonlinear problems. The direct and Newton-Raphson iterative methods were applied to the nonlinear equation of structural dynamics without any difficulty. As a nonlinear example, the highly nonlinear simple pendulum was solved by using the Newton-Raphson iterative method with various algorithms. For the highly nonlinear simple pendulum, two well-known second order algorithms (the Newmark method and the Baig and Bathe method) could not provide reasonable predictions even with very small time steps, while new higher-order algorithms could provide very accurate solutions even with considerably large size of the time step when solutions are compared with exactly obtained solution. In this particular problem, less computation time was taken for the new higher-order algorithms compared with the two second-order algorithms.

5. CONCLUSION

5.1 Summary and Concluding Remarks

In this dissertation, one family of second-order time integration algorithm and two families of higher-order time integration algorithms were developed. New time integration algorithms have been developed based on the time finite element method with unconventional computational framework. The main aim of this work has been to develop new families of time integration algorithms that could provide improved computational performance and functionality compared with existing second- and higher-order algorithms.

In Chapter 1, a review of existing second- and higher-order time integration algorithms was presented. The review included not only features of existing algorithms, but also the numerical methods which had been used to develop them. Recently developed time integration algorithms were reviewed and evaluated by using commonly used preferable attributes. In the review, some of existing time integration algorithms of major computational significance were selected, and both advantages and shortcomings of those selected algorithms were stated in detail, along with brief explanation of numerical methods used to develop them. The motivation and objective of this study have been stated at the end of the review.

Chapter 2 included a novel second-order time integration algorithm with unconditional stability, and controllable algorithmic dissipation. In this chapter some shortcomings of existing second-order algorithms were identified. To overcome stated shortcomings of existing second-order algorithms, the collocation method was applied to the unconventionally rewritten lower-order structural dynamic equation (called the unconventional mixed formulation). The sub-dividing strategy of the Baig and

Bathe method has been used, and collocation parameters has been considered for the mechanism of algorithmic dissipation control. New second-order time integration algorithm could include the Baig and Bathe method and the non-dissipative case as special cases within a single unified code. Some special traits of a parameter τ was also discussed. In new time integration algorithm, τ was used to improve the algorithmic damping ratio. Finally, the new algorithm was applied to the linear spring and the nonlinear FSDT problems. Solutions obtained from the new algorithm presented improved quality compared with the generalized-α method and the Baig and Bathe method.

Chapter 3 was one of two chapters devoted to the development of new higher-order time integration algorithms. A new family of higher-order algorithms has been developed based on the Hermite approximation in time and the modified weighted residual method. In this chapter, a modified weighted residual statement was introduced, and the integral form of the modified weighted residual statements were rewritten in algebraic forms by using weight parameters. After optimizing weight parameters to achieve improved accuracy and stability, the last remaining weight parameter was stated in terms of the ultimate spectral radius for the algorithmic dissipation control. General pth-order algorithms were obtained, if the pth-degree Hermite interpolation is used for the approximation of the displacement vector. The final forms of algorithms have been converted into condensed forms which can be readily implemented as computer code without any ambiguity. Especially, the elimination of the higher-order nodal time derivatives of the displacement was relatively easy due to the special computational structure of new algorithms. The special computational structure of new algorithms was obtained by including dynamic equilibriums and their time derivatives at the nodal points. As a result, unified $2m \times 2m$ form of condensed result equation was obtained, regardless of the degree of Hermite

approximation, m being the size of the semi-discrete system. Numerical examples were used to illustrate the advantages of using newly developed higher-order algorithms for spurious high frequency filtering. Accurate solutions were obtained in the long-term analysis by using newly developed higher-order algorithms.

In Chapter 4, another family of higher-order time integration algorithms which had been developed based on the unconventional mixed formulations and the modified weighted residual method were presented. Use of the mixed formulations allowed independent approximations for the displacement, velocity, and acceleration vectors, and the same Lagrange interpolation functions were used. $(2n-1)$th- and $(2n)$th order algorithms were obtained with nth-degree Lagrange interpolations. Two residual vectors were defined by using the time derivatives of the displacement-velocity and velocity-acceleration relations of the mixed formulations, then the weight parameters were used to rewrite the integral form of modified weighted residual statements into algebraic forms. Similar optimization procedure used in chapter 3 was also employed, and last remaining weight parameter was stated in terms of the ultimate spectral radius for a user specification type of algorithmic dissipation control. Obtained algorithms were fully extended to nonlinear cases, and specific nonlinear equation solving procedures were provided. Simple but specially considered numerical examples were used to demonstrate fundamental limitations of second-order algorithms, and advantages of newly developed higher-order algorithms. Numerical results confirmed that the proposed algorithms could provide similar level of accuracy of existing equivalent algorithms, while eliminating some computational shortcomings of them.

5.2 Future Works

In this dissertation several unconventional procedures for the development of improved implicit time integration algorithms for the analysis of structural dynamics

problems were proposed and extensively stated from a numerical performance viewpoint. New time integration algorithms have been analysed and tested to verify their performance and effectiveness. Since new algorithms were analysed and proved to be effective through this dissertation, they can be readily used for analyses of more complicated and challenging problems of structural dynamics.

The future applications would include highly interesting topics of structural dynamics such as (a) the transient analysis of the fluid-structure interaction problems, and (b) the impact and wave propagation problems in functionally graded materials and structures. The analysis and application of the algorithms presented in chapter 4 were limited to the second-order initial value problems (i.e., the equation of structural dynamics), and only brief guide lines of using them for the analyses of the first- and higher-order initial value problems have been provided. However, further researches regarding the extension of these algorithms to the first- and other higher-order (i.e., third- or higher-order) initial value problems should be conducted for the completeness of the study.

REFERENCES

[1] J. N. Reddy. *An introduction to the finite element method*. McGraw-Hill New York, 2006.

[2] D. L. Powers. *Boundary value problems: and partial differential equations*. Academic Press, 2009.

[3] S. J. Farlow. *Partial differential equations for scientists and engineers*. Courier Corporation, 2012.

[4] J. N. Reddy. *An Introduction to Nonlinear Finite Element Analysis: with applications to heat transfer, fluid mechanics, and solid mechanics*. Oxford, 2014.

[5] H. M. Hilber. *Analysis and design of numerical integration methods in structural dynamics*. PhD thesis, University of California Berkely, 1976.

[6] T. J. R. Hughes. *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation, 2012.

[7] J. Chung and G. M. Hulbert. A time integration algorithm for structural dynamics with improved numerical dissipation: The generalized-alpha method. *Journal of Applied Mechanics*, 60:271–275, 1993.

[8] K. J. Bathe and G. Noh. Insight into an implicit time integration scheme for structural dynamics. *Computers and Structures*, 98:1–6, 2012.

[9] J. C. Houbolt. A recurrence matrix solution for the dynamic response of aircraft in gusts. 1950.

[10] J. D. Kawamoto. *Solution of nonlinear dynamic structural systems by a hybrid frequency-time domain approach*. PhD thesis, Massachusetts Institute of Technology, 1983.

[11] C. Semler, W. C. Gentleman, P. M. P., et al. Numerical solutions of second order implicit non-linear ordinary differential equations. *Journal of Sound and Vibration*, 195(4):553–574, 1996.

[12] A. V. Idesman, H. Samajder, E. Aulisa, and P. Seshaiyer. Benchmark problems for wave propagation in elastic materials. *Computational Mechanics*, 43(6):797–814, 2009.

[13] A. V. Idesman. Accurate finite-element modeling of wave propagation in composite and functionally graded materials. *Composite Structures*, 117:298–308, 2014.

[14] K. J. Bathe. *Finite element procedures*. Klaus-Jurgen Bathe, 2006.

[15] O. C. Zienkiewicz and R. L. Taylor. *The finite element method*, volume 3. McGraw-hill London, 1977.

[16] O. C. Zienkiewicz and R. L. Taylor. *The finite element method for solid and structural mechanics*. Butterworth-heinemann, 2005.

[17] J. Chung. *Numerically dissipative time integration algorithms for structural dynamics*. PhD thesis, University of Michigan, 1992.

[18] S. Yin. A new explicit time integration method for structural dynamics. *International Journal of Structural Stability and Dynamics*, 13(03):1250068, 2013.

[19] I. Miranda, R. M. Ferencz, and T. J. R. Hughes. An improved implicit-explicit time integration method for structural dynamics. *Earthquake engineering & structural dynamics*, 18(5):643–653, 1989.

[20] H. M. Hilber, T. J. R. Hughes, and R. L. Taylor. Improved numerical dissipation for time integration algorithms in structural dynamics. *Earthquake Engineering and Structural Dynamics*, 5:283–292, 1977.

[21] G. M. Hulbert and T. J. R. Hughes. Space-time finite element methods for second-order hyperbolic equations. *Computer Methods in Applied Mechanics and Engineering*, 84(3):327–348, 1990.

[22] A. Cardona and M. Geradin. Time integration of the equations of motion in mechanism analysis. *Computers and Structures*, 33(3):801–820, 1989.

[23] N. M. Newmark. A method of computation for structural dynamics. *Journal of the Engineering Mechanics Division*, 85(3):67–94, 1959.

[24] E. L. Wilson. A computer program for the dynamic stress analysis of underground structures. Technical report, DTIC Document, 1968.

[25] K. J. Bathe and E. L. Wilson. Numerical methods in finite element analysis. 1976.

[26] K. C. Park. An improved stiffly stable method for direct integration of nonlinear structural dynamic equations. *Journal of Applied Mechanics*, 42(2):464–470, 1975.

[27] H. M Hilber and T. J. R. Hughes. Collocation, dissipation and [overshoot] for time integration schemes in structural dynamics. *Earthquake Engineering & Structural Dynamics*, 6(1):99–117, 1978.

[28] H. M. Hilber, T. J. R. Hughes, and R. L. Taylor. Improved numerical dissipation for time integration algorithms in structural dynamics. *Earthquake Engineering & Structural Dynamics*, 5(3):283–292, 1977.

[29] G. Bazzi and E. Anderheggen. The ρ-family of algorithms for time-step integration with improved numerical dissipation. *Earthquake Engineering & Structural Dynamics*, 10(4):537–550, 1982.

[30] W. L. Wood. *Practical time-stepping schemes*. Oxford University Press, USA, 1990.

[31] O. C. Zienkiewicz and R. L. Taylor. The finite element method, 1991.

[32] C. Hoff and P. J. Pahl. Development of an implicit method with numerical dissipation from a generalized single-step algorithm for structural dynamics. *Computer Methods in Applied Mechanics and Engineering*, 67(3):367–385, 1988.

[33] T. C. Fung. Numerical dissipation in time-step integration algorithms for structural dynamic analysis. *Progress in Structural Engineering and Materials*, 5(3):167–180, 2003.

[34] W. L. Wood, M. Bossak, and O. C. Zienkiewicz. An alpha modification of newmark's method. *International Journal of Numerical Methods in Engineering*, 15:1562–1566, 1981.

[35] M. M. I. Baig and K. J. Bathe. On direct time integration in large deformation dynamic analysis. In *3rd MIT conference on computational fluid and solid mechanics*, pages 1044–1047, 2005.

[36] G. M. Hulbert. A unified set of single-step asymptotic annihilation algorithms for structural dynamics. *Computer Methods in Applied Mechanics and Engineering*, 113(1):1–9, 1994.

[37] S. J. Kim, J. Y. Cho, and W. D. Kim. From the trapezoidal rule to higher-order accurate and unconditionally stable time-integration method for structural dynamics. *Computer Methods in Applied Mechanics and Engineering*, 149(1):73–88, 1997.

[38] T. C. Fung. Complex-time-step newmark methods with controllable numerical dissipation. *International Journal for numerical methods in Engineering*,

41(1):65–93, 1998.

[39] C. M. Shearer and C. E. S. Cesnik. *Modified Generalized-Alpha Method for Integrating Governing Equations of Very Flexible Aircraft*. Defense Technical Information Center, 2006.

[40] K. J. Bathe and M. M. I. Baig. On a composite implicit time integration procedure for nonlinear dynamics. *Computers and Structures*, 83(31):2513–2524, 2005.

[41] K. J. Bathe. Conserving energy and momentum in nonlinear dynamics: a simple implicit time integration scheme. *Computers and Structures*, 85(7):437–445, 2007.

[42] R. Al-Khoury, J. Weerheijm, K. Dingerdis, and L. J. Sluys. An adaptive time integration scheme for blast loading on a saturated soil mass. *Computers and Geotechnics*, 38(4):448–464, 2011.

[43] A. V. Idesman. A new high-order accurate continuous galerkin method for linear elastodynamics problems. *Computational Mechanics*, 40(2):261–279, 2007.

[44] T. C. Fung. Solving initial value problems by differential quadrature method-part 2: second-and higher-order equations. *International Journal for Numerical Methods in Engineering*, 50(6):1429–1454, 2001.

[45] D. Kuhl and M. A. Crisfield. Energy-conserving and decaying algorithms in non-linear structural dynamics. *International journal for numerical methods in engineering*, 45(5):569–599, 1999.

[46] S. Erlicher, L. Bonaventura, and O. S. Bursi. The analysis of the generalized-α method for non-linear dynamic problems. *Computational Mechanics*, 28(2):83–104, 2002.

[47] A. Bonelli, O. S. Bursi, S. Erlicher, and L. Vulcan. Analyses of the generalized-α method for linear and non-linear forced excited systems. In *Structural Dynamics-EURODYN*, volume 2, pages 1523–1528. Citeseer, 2002.

[48] G. M. Hulbert. Limitations on linear multistep methods for structural dynamics. *Earthquake Engineering & Structural Dynamics*, 20(2):191–196, 1991.

[49] T. J. R. Hughes. Analysis of transient algorithms with particular reference to stability behavior. *In Computational Methods for Transient Analysis*, 1:67–155, 1983.

[50] M. Austin. High-order integration of smooth dynamical systems: Theory and numerical experiments. *International journal for numerical methods in engineering*, 36(12):2107–2122, 1993.

[51] N. Tarnow and J. C. Simo. How to render second order accurate time-stepping algorithms fourth order accurate while retaining the stability and conservation properties. *Computer Methods in Applied Mechanics and Engineering*, 115(3):233–252, 1994.

[52] T. C. Fung. Unconditionally stable higher-order newmark methods by substepping procedure. *Computer Methods in Applied Mechanics and Engineering*, 147(1):61–84, 1997.

[53] T. C. Fung. Higher order time-step integration methods with complex time steps. *Journal of sound and vibration*, 210(1):69–89, 1998.

[54] I. Fried. Finite-element analysis of time-dependent phenomena. *AIAA Journal*, 7(6):1170–1173, 1969.

[55] M. Geradin. On the variational method in the direct integration of the transient structural response. *Journal of Sound and Vibration*, 34(4):479–487, 1974.

[56] R. Riff and M. Baruch. Time finite element discretization of hamilton's law of varying action. *AIAA journal*, 22(9):1310–1318, 1984.

[57] C. D. Bailey. Further remarks on the law of varying action and the symbol δ. *Journal of Sound and Vibration*, 131(2):331–344, 1989.

[58] J. N. Reddy. *Energy principles and variational methods in applied mechanics*. John Wiley & Sons, 2002.

[59] T. C. Fung. Unconditionally stable higher-order accurate hermitian time finite elements. *International Journal for Numerical Methods in Engineering*, 39(20):3475–3495, 1996.

[60] T. C. Fung. Weighting parameters for unconditionally stable higher-order accurate time step integration algorithms. part 2-second-order equations. *International journal for numerical methods in engineering*, 45(8):971–1006, 1999.

[61] B. W. Golley and M. Amer. An unconditionally stable time-stepping procedure with algorithmic damping: a weighted integral approach using two general weight functions. *Earthquake engineering & structural dynamics*, 28(11):1345–1360, 1999.

[62] J. H. Argyris, L. E. Vaz, and K. J. Willam. Higher order methods for transient diffusion analysis. *Computer Methods in Applied Mechanics and Engineering*, 12(2):243–278, 1977.

[63] B. W. Golley. A time-stepping procedure for structural dynamics using gauss point collocation. *International journal for numerical methods in engineering*, 39(23):3985–3998, 1996.

[64] T. C. Fung. Unconditionally stable collocation algorithms for second order initial value problems. *Journal of sound and vibration*, 247(2):343–365, 2001.

[65] S. Rostami, S. Shojaee, and A. Moeinadini. A parabolic acceleration time integration method for structural dynamics using quartic b-spline functions. *Applied Mathematical Modelling*, 36(11):5162–5182, 2012.

[66] G. R. Liu and T. Y. Wu. Numerical solution for differential equations of duffing-type non-linearity using the generalized differential quadrature rule. *Journal of Sound and Vibration*, 237(5):805–817, 2000.

[67] G. R. Liu and T. Y. Wu. Vibration analysis of beams using the generalized differential quadrature rule and domain decomposition. *Journal of Sound and Vibration*, 246(3):461–481, 2001.

[68] J. Liu and X. Wang. An assessment of the differential quadrature time integration scheme for nonlinear dynamic equations. *Journal of Sound and Vibration*, 314(1):246–253, 2008.

[69] T. C. Fung. Solving initial value problems by differential quadrature method-part 1: first-order equations. *International Journal for Numerical Methods in Engineering*, 50(6):1411–1427, 2001.

[70] T. C. Fung. On the equivalence of the time domain differential quadrature method and the dissipative runge–kutta collocation method. *International journal for numerical methods in engineering*, 53(2):409–431, 2002.

[71] T. C. Fung. Unconditionally stable collocation algorithms for second order initial value problems. *Journal of sound and vibration*, 247(2):343–365, 2001.

[72] T. C. Fung. Weighting parameters for unconditionally stable higher-order accurate time step integration algorithms. part 1-first-order equations. *International Journal for Numerical Methods in Engineering*, 45(8):941–970, 1999.

[73] J. Har and K. Tamma. *Advances in computational dynamics of particles, materials and structures*. John Wiley & Sons, 2012.

[74] J. Ma. *A new space-time finite element method for the dynamic analysis of truss-type structures*. PhD thesis, Edinburgh Napier University, 2015.

[75] N. S. Putcha and J. N. Reddy. A refined mixed shear flexible finite element for the nonlinear analysis of laminated plates. *Computers and Structures*, 22(4):529–538, 1986.

[76] W. Kim. Unconventional finite element models for nonlinear analysis of beams and plates, 2008.

[77] W. Kim and J. N. Reddy. Novel mixed finite element models for nonlinear analysis of plates. *Latin American Journal of Solids and Structures*, 7(2):201–226, 2010.

[78] A. V. Idesman, M. Schmidt, and R. L. Sierakowski. A new explicit predictor–multicorrector high-order accurate method for linear elastodynamics. *Journal of Sound and Vibration*, 310(1):217–229, 2008.

[79] O. C. Zienkiewicz, R. L. Taylor, and J.Z. Zhu. *The finite element method: its basis and fundamentals*. Butterworth-Heinemann Burlington, VT, 2005.

[80] R. Bellman and J. Casti. Differential quadrature and long-term integration. *Journal of Mathematical Analysis and Applications*, 34(2):235–238, 1971.

[81] D. Kuhl and E. Ramm. Constraint energy momentum algorithm and its application to non-linear dynamics of shells. *Computer Methods in Applied Mechanics and Engineering*, 136(3):293–315, 1996.

[82] W Kim, S. Park, and J. N. Reddy. A cross weighted-residual time integration scheme for structural dynamics. *International Journal of Structural Stability and Dynamics*, 14(06):1450023, 2014.

[83] W. Kim and J.N. Reddy. A comparative study of least-squares and the weak-form galerkin finite element models for the nonlinear analysis of timoshenko beams. *Journal of Solid Mechanics*, 2(2):101–114, 2010.

[84] J. T. Oden. A general theory of finite elements ii. applications. *International Journal for Numerical Methods in Engineering*, 1(3):247–259, 1969.

[85] J. Argyris and H. P. Mlejnek. Dynamics of structures. texts on computational mechanics. vol. 5. *North-Holland*, 1991.

[86] K. M. Singh and M. S. Kalra. Least-squares finite element schemes in the time domain. *Computer methods in applied mechanics and engineering*, 190(1):111–131, 2000.

[87] J. T. Oden and J. N. Reddy. *An introduction to the mathematical theory of finite elements*. Courier Corporation, 2012.

[88] G. M. Hulbert and J. Chung. The unimportance of the spurious root of time integration algorithms for structural dynamics. *Communications in numerical methods in engineering*, 10(8):591–597, 1994.

[89] T. J. R. Hughes and T. E. Tezduyar. Stability and accuracy analysis of some fully-discrete algorithms for the one-dimensional second-order wave equation. *Computers and Structures*, 19:665–668, 1984.

[90] K. J. Bathe. Frontiers in finite element procedures & applications. *Computational Methods for Engineering Science*, 2014.

[91] S. J. Farlow. *Partial Differential Equations for Scientists and Engineers*. Wiley, New York, 1993.

[92] G. F. Howard and J. Penny. The accuracy and stability of time domain finite element solutions. *Journal of Sound and Vibration*, 61(4):585–595, 1978.

[93] M. Gellert. A new algorithm for integration of dynamic systems. *Computers and Structures*, 9(4):401–408, 1978.

[94] S. J. Leon. *Linear algebra with applications*. Macmillan New York, 1980.

[95] T. C. Fung and S. K. Chow. Solving non-linear problems by complex time step methods. *Communications in numerical methods in engineering*, 18(4):287–303, 2002.

[96] I. Fried and D. S. Malkus. Finite element mass matrix lumping by numerical integration with no convergence rate loss. *International Journal of Solids and Structures*, 11(4):461–466, 1975.

[97] D. Schillinger, J. A. Evans, F. Frischmann, R. R. Hiemstra, M. Hsu, and T. J. R. Hughes. A collocated c0 finite element method: Reduced quadrature perspective, cost comparison with standard finite elements, and explicit structural dynamics. *International Journal for Numerical Methods in Engineering*, 102(3-4):576–631, 2015.