 CMES, vol.104, no.2, pp.133-158, 2015

The Fundamentals Underlying the Computations of Acceleration for General Dynamic Applications: Issues and Noteworthy Perspectives

M. Shimada[1], **A. Hoitink**[1] **and K. K. Tamma**[1,2]

Abstract: To-date, with the exception of the Newmark method and the midpoint rule, most computational algorithms under the umbrella of LMS methods, which are predominantly employed in research and commercial software fail to properly evaluate acceleration computations accurately for conducting the numerical dynamic simulations. Indeed, this is not trivial and a sound theoretical basis of the fundamental underlying issues is described in detail. In this paper, we provide a resolution and point-out several noteworthy perspectives to address the proper evaluation of acceleration computations for structural dynamics applications with focus on the class of LMS methods as an illustration.

1 Introduction

The present paper describes the accurate and precise evaluation of acceleration computations which play an important role in general structural dynamics applications. Computational algorithms for time dependent problems are mostly designed after the semi-discretization is done on the partial differential equations governing the equations of motion. In particular, these computational algorithms are predominantly the class of linear multi-step (LMS) methods that are widely used for general engineering applications and most commercial software. Numerous research efforts have been undertaken in designing computational algorithms over the past fifty years or so starting with the Houbolt method, Newmark based methods and the like. The two principal classes of algorithms are numerically non-dissipative methods such as Newmark, the velocity based scheme, the classical midpoint rule, and controllable numerical dissipative methods to control high frequency behavior encountered in stiff dynamical systems. It is well known that numerous application areas exist in science and engineering wherein acceleration calculations need

[1] Department of Mechanical Engineering, University of Minnesota, 111 Church St. SE, Minneapolis, MN, 55455, USA.
[2] Corresponding Author. E-mail: ktamma@umn.edu

to be precise. Typical applications encompass determination of g-forces encountered in dynamical systems to include impact applications where it is extremely useful to have a knowledge of the accelerations imparted at critical sections of vehicle structures carrying sensitive electronics, in the evaluation of base accelerations imparted due to missiles/torpedos and the like in the vicinity of ship structures which is critical for assessments of ship dynamics, earthquake engineering applications involving base ground motion, and the like. The fundamental question that is posed regarding all these developments to-date is: "are the accelerations that are being computed accurate and proper"? Alternatively, this paper presents the exsiting drawbacks and provides a fundamental resolution to the accurate and proper computation of accelerations for both numerically non-dissipative and dissipative methods with a clear resolution for the entire class of LMS methods. Consequently, future developments can now accurately implement the proper dynamics for general engineering applications precisely. For illustration, the focus of this paper is on the class of LMS methods which are most popular in research and commercial software.

Recently, Zhou and Tamma Tamma, Zhou, and Sha (2000); Tamma, Kanapady, Zhou, and Sha (2000); Zhou and Tamma (2004b,a, 2006) described a unified theory underlying computational algorithms for designing computational algorithms for time dependent phenomenon. In particular, focusing attention on the class of LMS methods for general structural dynamics applications which are the predominant methods in most commercial software, Zhou and Tamma described a general framework under the umbrella of generalized single step single solve [GSSSS] family of algorithms. This novel framework encompasses most of the LMS methods that have been developed over the past fifty years or so, and also additionally includes new and optimal methods that have not been available to-date that are optimal for various applications in structural dynamics. A brief highlight of the so-called big picture follows next so as to put into context the subtle issues and the underlying resolution of the existing drawbacks and deficiencies prevalent in acceleration computations to-date. The GSSSS family of algorithms can be viewed as the fully discretized time integration algorithms for the equation of motion which contains most past and also new and recent computational developments. The novelty of this framework is that a single precise time integration module is all that is needed to be implemented which provides all possible algorithms as available choices and options for the analyst to use. Basically, within this framework there exist two distinct families of algorithms termed as constrained U (displacement-overshoot aspects) and constrained V (velocity overshoot aspects) family of algorithms. They simply characterize the overshoot behavior on the displacement or the velocity fields of a particular algorithm. For example, the Newmark Newmark

(1959) algorithm which is a numerically non-dissipative method belongs to the U0 family with particular additional characteristics termed as V0; that is it does not have any overshoot behavior for the displacement or velocity. Alternately, the so-called HHT-αHilber, Hughes, and Taylor (1977), is a numerically controllable dissipative method that was introduced to control high frequency behavior and it is a U0V1 algorithm belonging to the U0 family (same as Newmark), which is characterized by no overshoot in displacement but inherits first order overshoot in the velocity field. These overshoot characteristics play an important role in that they enter the computations; consequently, higher overshoot behavior may cause nonlinear iterations in implicit dynamics calculations to not converge. There exist numerous algorithms that are contained within the GSSSS family including other more recently developed optimal algorithms with and without controllable numerical dissipation that are preferable for many engineering applications Zhou and Tamma (2004b).

The GSSSS Family of Algorithms: Let the time interval of interest, $T = t_L - t_0$, be divided into n_t sub-intervals such that $\mathbb{I} = \bigcup_{n=0}^{\bar{n}-1}[t_n, t_{n+1}]$, assuming $t_0 < t_1 < \cdots < t_{\bar{n}} \equiv t_L$. The time step size is defined as $\Delta t := t_{n+1} - t_n$. Suppose the initial conditions $(\mathbf{q}, \dot{\mathbf{q}})(t_0) = (\mathbf{q}_0, \mathbf{v}_0)$[1] are given; then, we can compute the numerical displacement (configuration) $\mathbf{q}_{n+1}$, velocity $\mathbf{v}_{n+1}$, and acceleration $\mathbf{a}_{n+1}$ for $n \in \{0, 1, 2, \cdots, \bar{n}-1\}$ from:

$$\mathbf{M}\tilde{\mathbf{a}} = \mathbf{F}(\tilde{\mathbf{q}}, \tilde{\mathbf{v}}, t_{n+W_1}) \tag{1}$$

where $t_{n+W_1} := (1 - W_1)t_n + W_1 t_{n+1}$, and the algorithmic configuration, velocity, and acceleration vectors are given by

$$\tilde{\mathbf{q}} = \mathbf{q}_n + W_1\Lambda_1\mathbf{v}_n\Delta t + W_2\Lambda_2\mathbf{a}_n\Delta t^2 + W_3\Lambda_3\Delta\mathbf{a}\Delta t^2\eta_2 \tag{2}$$

$$\tilde{\mathbf{v}} = \mathbf{v}_n + W_1\Lambda_4\mathbf{a}_n\Delta t + W_2\Lambda_5\Delta\mathbf{a}\Delta t\eta_1 \tag{3}$$

$$\tilde{\mathbf{a}} = \mathbf{a}_n + W_1\Lambda_6\Delta\mathbf{a} \tag{4}$$

And the corresponding updates are designed to be

$$\begin{aligned}
\mathbf{q}_{n+1} &= \mathbf{q}_n + \lambda_1\mathbf{v}_n\Delta t + \lambda_2\mathbf{a}_n\Delta t^2 + \lambda_3\Delta\mathbf{a}\Delta t^2\eta_3 \\
\mathbf{v}_{n+1} &= \mathbf{v}_n + \lambda_4\mathbf{a}_n\Delta t + \lambda_5\Delta\mathbf{a}\Delta t \\
\mathbf{a}_{n+1} &= \mathbf{a}_n + \Delta\mathbf{a}
\end{aligned} \tag{5}$$

The algorithmic scalar parameters are given by

$$W_1 = W_2 = W_3 = \frac{1}{1+\rho_\infty^s}, \quad \lambda_1 = \Lambda_1 = \lambda_4 = \Lambda_4 = 1$$

[1] $\dot{\square}$ denotes the time derivative of $\square$. And $\square_n$ denotes the numerical value, from the GSSSS family of algorithms, at time t_n. For example, $\mathbf{v}_n$ is the numerical value of $\dot{\mathbf{q}}(t_n)$, the velocity $\dot{\mathbf{q}}$ at time t_n, i.e., $\mathbf{v}_n \approx \dot{\mathbf{q}}(t_n)$.

$$\lambda_2 = \Lambda_2 = \frac{1}{2}, \quad \lambda_3 = \Lambda_3 = \frac{1}{(1+\rho_\infty^{\min})(1+\rho_\infty^{\max})}$$

$$W_1\Lambda_6 = \frac{2+\rho_\infty^{\min}+\rho_\infty^{\max}+\rho_\infty^{\rm s}-\rho_\infty^{\min}\rho_\infty^{\max}\rho_\infty^{\rm s}}{(1+\rho_\infty^{\min})(1+\rho_\infty^{\max})(1+\rho_\infty^{\rm s})},$$

$$\lambda_5 = \Lambda_5 = \frac{3+\rho_\infty^{\min}+\rho_\infty^{\max}-\rho_\infty^{\min}\rho_\infty^{\max}}{2(1+\rho_\infty^{\min})(1+\rho_\infty^{\max})}$$

for the ***U0 family-based schemes***, and

$$W_1 = \frac{3+\rho_\infty^{\min}+\rho_\infty^{\max}-\rho_\infty^{\min}\rho_\infty^{\max}}{2(1+\rho_\infty^{\min})(1+\rho_\infty^{\max})}, \quad \lambda_1 = \Lambda_1 = \lambda_4 = \Lambda_4 = 1$$

$$W_2 = W_3 = \frac{2}{(1+\rho_\infty^{\min})(1+\rho_\infty^{\max})}, \quad \lambda_2 = \Lambda_2 = \frac{1}{2}$$

$$\lambda_3 = \Lambda_3 = \frac{1}{2(1+\rho_\infty^{\rm s})}, \quad \lambda_5 = \Lambda_5 = \frac{1}{1+\rho_\infty^{\rm s}}$$

$$W_1\Lambda_6 = \frac{2+\rho_\infty^{\min}+\rho_\infty^{\max}+\rho_\infty^{\rm s}-\rho_\infty^{\min}\rho_\infty^{\max}\rho_\infty^{\rm s}}{(1+\rho_\infty^{\min})(1+\rho_\infty^{\max})(1+\rho_\infty^{\rm s})}$$

for the ***V0 family-based schemes***. The additional algorithmic parameters η_i (for $i=1,2,3$) govern the various features of the corresponding explicit members within this general algorithmic framework for second-order systems. For the implicit case ($\eta_i = 1$ for $i=1,2,3$), the roots must satisfy the following relation:

$$0 \le \rho_\infty^{\rm s} \le \rho_\infty^{\min} \le \rho_\infty^{\max} \le 1 \tag{6}$$

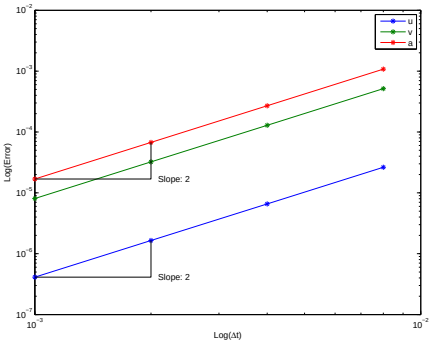
For linear dynamical systems, the right-hand side of Eq. (1) may yield

$$\mathbf{F}(\tilde{\mathbf{q}}, \tilde{\mathbf{v}}, t_{n+W_1}) = -\mathbf{C}\tilde{\mathbf{v}} - \mathbf{K}\tilde{\mathbf{q}} + \mathbf{f}(t_{n+W_1}) \tag{7}$$

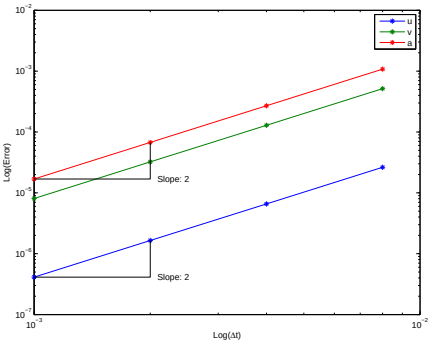
where $\mathbf{C}$ and $\mathbf{K}$ denote the damping and stiffness matrices, respectively; and $\mathbf{f}(t)$ denotes the time-dependent external load vector.

The Issue: While the displacement and velocity vectors are indeed evaluated precisely at the time level $t = t_{n+1}$, the acceleration computations that are reported in the literature and the like, are, in general, not accurately determined and/or are being misinterpreted by the research and commercial computational community at large. For example while it is true (actually happens to be true) for the implicit Newmark method that $\mathbf{a}_{n+1}$ indeed is accurately computed at time level $n+1$, for most of the other methods this is not the general case. As a demonstration, Fig.

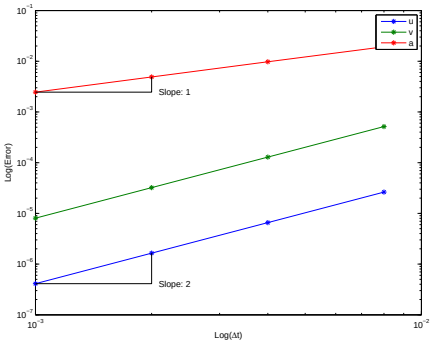
1 shows convergence plots using the traditional (misinterpreted) method for the single degree of freedom problem described later in the numerical examples. We see that for the implicit Newmark method and the classical midpoint rule (numerically non-dissipative methods) that the order of convergence for all three variables is indeed two. For these two special cases, it is indeed true that the accelerations resulting from the algorithm falls at the time $n+1$. However, for the particular numerically non-dissipative velocity based scheme Tamma and Namburu (1990) in the sense of LMS methods, this is not true (the acceleration comes out to be first order only). For all other algorithms, including the numerically non-dissipative velocity based scheme shown, the acceleration time level resulting from the algorithm is "shifted" from displacement and velocity time levels, and thus incorrectly yields only first order accurate results.



(a) U0V0(1,1,0): Implicit Newmark



(b) U0V0/V0U0(1,1,1): MPR-EPA



(c) V0U0(1,1,0): MPR-MPA

Figure 1: Traditional convergence of algorithms without dissipation (Notice that the acceleration is incorrectly shown to come out to be first-order for the MPR-MPA algorithm)

It is also routinely shown in the literature that for algorithms which include controllable numerical dissipation, the order of accuracy in the acceleration is less than two. Fig. 2 illustrates just one example of this using for example, the U0V1 optimal scheme, which is identical to the so-called three parameter optimal method or its equivalent, namely, the generalized-α method (and is taken from Ref. Erlicher, Bonaventura, and Bursi (2002)).

There exists a fundamental misconception in the use and interpretation of the term "acceleration and the time level at which it is interpreted and computed;" and this paper provides a deep insight and a resolution to the issue, within the entire LMS class of algorithms chosen simply for illustration of the basic ideas.

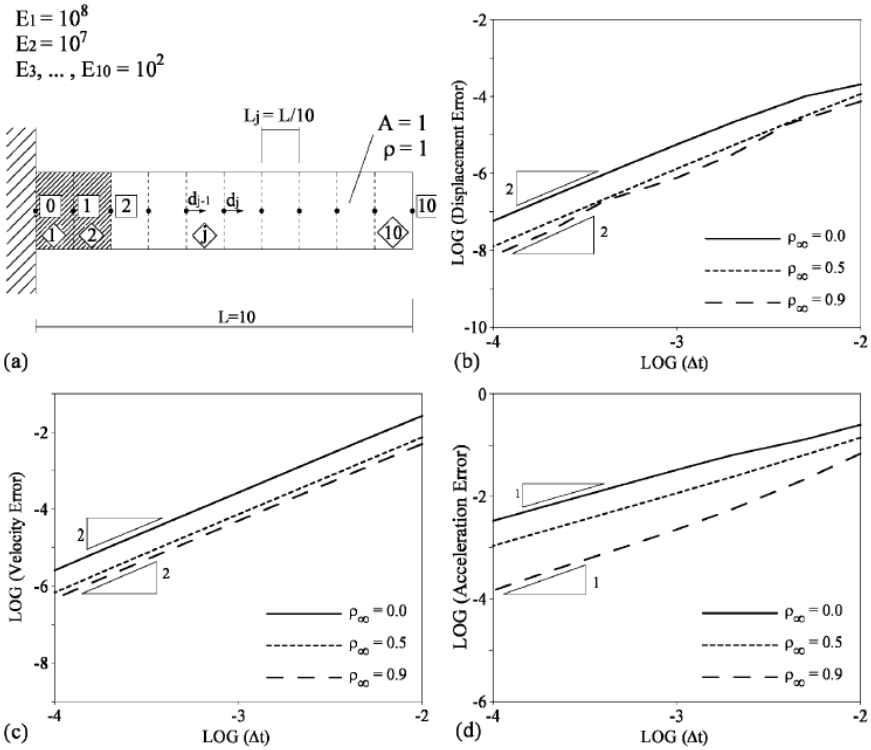


Figure 2: Figures from Ref Erlicher, Bonaventura, and Bursi (2002) showing the displacement and velocity second order time accurate, but the acceleration is only cited to be first order time accurate using the U0V1 optimal scheme, which is identical to the three parameter optinal method or its equivalent, namely, the generalized-α method

Understanding the time level at which the acceleration is computed in general, for the class of computational algorithms in the sense of the LMS methods is the key to showing proper accuracy. Again, the fundamental concept is that, in the general case of computational algorithms, the acceleration is strictly *not* being calculated at time level t_{n+1}. How and where the acceleration time level is calculated and how to describe an accurate convergence plot which shows correct accuracy is discussed at length below based for the framework of generalized single step single solve (GSSSS) algorithms developed by Zhou and Tamma Zhou and Tamma (2004b) for the entire class of LMS methods that are predominant in most research and commercial software.

2 Time Level Analysis of the GSSSS Family of Algorithms

Both the U0 and V0 based-family of algorithms shown above have been particularly designed to be of second-order time accuracy in the displacement, velocity, and external load vectors; however, we get only first-order accuracy in the acceleration vector if we assume $\mathbf{a}_n \approx \ddot{\mathbf{q}}(t_n)$ and $\mathbf{a}_{n+1} \approx \ddot{\mathbf{q}}(t_{n+1})$. In order to guarantee the second-order time accuracy in all the variables, one must be aware of the correct time level of approximations for $\mathbf{q}$, $\dot{\mathbf{q}}$, and $\ddot{\mathbf{q}}$ as

$$\begin{aligned} &\mathbf{q}_n \approx \mathbf{q}(t_n) \ \text{ and } \ \mathbf{q}_{n+1} \approx \mathbf{q}(t_{n+1}) \\ &\mathbf{v}_n \approx \dot{\mathbf{q}}(t_n) \ \text{ and } \ \mathbf{v}_{n+1} \approx \dot{\mathbf{q}}(t_{n+1}) \\ &\mathbf{a}_n \approx \ddot{\mathbf{q}}(t_{n-\phi}) \ \text{ and } \ \mathbf{a}_{n+1} \approx \ddot{\mathbf{q}}(t_{n+1-\phi}) \end{aligned} \tag{8}$$

where $\phi := W_1(\Lambda_6 - 1) \in \mathbb{R}$.

Theorem 1 (Time Level of Acceleration)
In the U0 and V0 based-family of algorithms, the second-order time accuracy in $\ddot{\mathbf{q}}$, *i.e.,*

$$\mathbf{a}_{n+1} - \ddot{\mathbf{q}}(t_{n+1-\phi}) = \mathscr{O}(\Delta t^2) \ \textit{ with } \ \phi := W_1(\Lambda_6 - 1) \tag{9}$$

is obtained for a solution satisfying

$$\begin{aligned} &\mathbf{q}_n - \mathbf{q}(t_n) = \mathscr{O}(\Delta t^2) \ \textit{ and } \ \mathbf{q}_{n+1} - \mathbf{q}(t_{n+1}) = \mathscr{O}(\Delta t^2) \\ &\mathbf{v}_n - \dot{\mathbf{q}}(t_n) = \mathscr{O}(\Delta t^2) \ \textit{ and } \ \mathbf{v}_{n+1} - \dot{\mathbf{q}}(t_{n+1}) = \mathscr{O}(\Delta t^2) \end{aligned} \tag{10}$$

if $\mathbf{a}_n - \ddot{\mathbf{q}}(t_{n-\phi}) = \mathscr{O}(\Delta t^2)$ *is guaranteed.*

Proof. In this proof, we only consider the V0 family-based algorithms since the proof for the U0 family-based algorithms is more straightforward, and we get the

same result as stated in the theorem above. The V0 family-based algorithms can be cast into

$$\mathbf{M}\tilde{\mathbf{a}} + \mathbf{C}\tilde{\mathbf{v}} + \mathbf{K}\tilde{\mathbf{q}} = \tilde{\mathbf{f}} \tag{11}$$

where the algorithmic $\ddot{\mathbf{q}}$, $\dot{\mathbf{q}}$, $\mathbf{q}$, and $\mathbf{f}$ are given by

$$\begin{aligned}
\tilde{\mathbf{a}} &= (1 - W_1\Lambda_6)\mathbf{a}_n + W_1\Lambda_6\mathbf{a}_{n+1} && (12)\\
\tilde{\mathbf{v}} &= \mathbf{v}_n + W_2(\mathbf{v}_{n+1} - \mathbf{v}_n) + \Delta t(W_1 - W_2)\mathbf{a}_n && (13)\\
\tilde{\mathbf{q}} &= \mathbf{q}_n + W_2(\mathbf{q}_{n+1} - \mathbf{q}_n) + \Delta t(W_1 - W_2)\mathbf{v}_n && (14)\\
\tilde{\mathbf{f}} &= (1 - W_1)\mathbf{f}_n + W_1\mathbf{f}_{n+1} && (15)
\end{aligned}$$

Using Equation (10), we get

$$\begin{aligned}
\tilde{\mathbf{v}} &= \dot{\mathbf{q}}(t_n) + W_2[\dot{\mathbf{q}}(t_{n+1}) - \dot{\mathbf{q}}(t_n)] + \Delta t(W_1 - W_2)\ddot{\mathbf{q}}(t_{n-\phi}) + \mathscr{O}(\Delta t^p)\\
&= \dot{\mathbf{q}}(t_n) + \Delta t W_1\ddot{\mathbf{q}}(t_n) + \mathscr{O}(\Delta t^p) && (16)
\end{aligned}$$

where $p = 2$ and $p = 1$ for $s \geq 2$ and $s = 1$, respectively, in $\mathbf{a}_n - \ddot{\mathbf{q}}(t_{n-\phi}) = \mathscr{O}(\Delta t^s)$. Similarly,

$$\begin{aligned}
\tilde{\mathbf{q}} &= \mathbf{q}(t_n) + W_2[\mathbf{q}(t_{n+1}) - \mathbf{q}(t_n)] + \Delta t(W_1 - W_2)\dot{\mathbf{q}}(t_{n-\phi}) + \mathscr{O}(\Delta t^2)\\
&= \mathbf{q}(t_n) + \Delta t W_1\dot{\mathbf{q}}(t_n) + \mathscr{O}(\Delta t^2) && (17)
\end{aligned}$$

and

$$\tilde{\mathbf{f}} = \mathbf{f}(t_n) + W_1[\mathbf{f}(t_{n+1}) - \mathbf{f}(t_n)] + \mathscr{O}(\Delta t^2) = \mathbf{f}(t_n) + \Delta t W_1\dot{\mathbf{f}}(t_n) + \mathscr{O}(\Delta t^2) \tag{18}$$

Since the Taylor series expansions about time $t = t_n$ yields

$$\begin{aligned}
\ddot{\mathbf{q}}(t_{n+1-\phi}) &= \ddot{\mathbf{q}}(t_n) + (1 - \phi)\Delta t\dddot{\mathbf{q}}(t_n) + \mathscr{O}(\Delta t^2)\\
\ddot{\mathbf{q}}(t_{n-\phi}) &= \ddot{\mathbf{q}}(t_n) - \phi\Delta t\dddot{\mathbf{q}}(t_n) + \mathscr{O}(\Delta t^2)
\end{aligned} \tag{19}$$

we get

$$(1 - W_1\Lambda_6)\ddot{\mathbf{q}}(t_{n-\phi}) + W_1\Lambda_6\ddot{\mathbf{q}}(t_{n+1-\phi}) = \ddot{\mathbf{q}}(t_n) + \Delta t(W_1\Lambda_6 - \phi)\dddot{\mathbf{q}}(t_n) + \mathscr{O}(\Delta t^2) \tag{20}$$

From Equations (11)-(20) with $\ddot{\mathbf{q}}(t) = -\mathbf{M}^{-1}[\mathbf{C}\dot{\mathbf{q}}(t) + \mathbf{K}\mathbf{q}(t) - \mathbf{f}(t)]$,

$$\begin{aligned}
&(1 - W_1\Lambda_6)[\mathbf{a}_n - \ddot{\mathbf{q}}(t_{n-\phi})] + W_1\Lambda_6[\mathbf{a}_{n+1} - \ddot{\mathbf{q}}(t_{n+1-\phi})]\\
&= \Delta t(W_1 - W_1\Lambda_6 + \phi)\dddot{\mathbf{q}}(t_n) + \mathscr{O}(\Delta t^p)
\end{aligned} \tag{21}$$

Hence, $\mathbf{a}_{n+1} - \ddot{\mathbf{q}}(t_{n+1-\phi}) = \mathscr{O}(\Delta t^2)$ is obtained when $\mathbf{a}_n - \ddot{\mathbf{q}}(t_{n-\phi}) = \mathscr{O}(\Delta t^s)$ with $s \geq 2$ for $\phi = W_1(\Lambda_6 - 1)$.

Remark 2.1 (Theorem 1)

1. *Choosing the initial value of* $\ddot{\mathbf{q}}$ *as* $\mathbf{a}_0 = \ddot{\mathbf{q}}(t_0) = -\mathbf{M}^{-1}[\mathbf{C}\dot{\mathbf{q}}(t_0) + \mathbf{K}\mathbf{q}(t_0) - \mathbf{f}(t_0)]$, *actually still yields the second-order accuracy in* $\ddot{\mathbf{q}}$.

2. *It is important to note that* $\mathbf{a}_n \approx \ddot{\mathbf{q}}(t_{n-\phi})$ *and* $\mathbf{a}_{n+1} \approx \ddot{\mathbf{q}}(t_{n+1-\phi})$ *are not the approximations of* $\ddot{\mathbf{q}}$ *at time* $t = t_n$ *and* $t = t_{n+1}$, *respectively. Therefore, to plot the* ***true acceleration*** *time history, one must use*

$$\ddot{\mathbf{q}}(t_1) \approx \frac{1}{1-\phi}\mathbf{a}_1 - \frac{\phi}{1-\phi}\mathbf{a}_0 =: \hat{\mathbf{a}}_1 \tag{22}$$

at the first time step and

$$\ddot{\mathbf{q}}(t_{n+1}) \approx (1+\phi)\mathbf{a}_{n+1} - \phi\mathbf{a}_n =: \hat{\mathbf{a}}_{n+1} \tag{23}$$

for $n \in \{1, 2, \cdots, n_t - 1\}$. *This is illustrated in Fig. 3.*

3. *The spectral condition V0*$\{1.0, 1.0, \rho_\infty^{\mathrm{s}}\}$ *with* ρ_∞^{s} *in the V0 family-based algorithms leads to*

$$\mathbf{M}\tilde{\mathbf{a}} + \mathbf{C}\left[\mathbf{v}_n + \frac{\Delta t}{2}\tilde{\mathbf{a}}\right] + \mathbf{K}\left[\mathbf{q}_n + \frac{\Delta t}{2}\mathbf{v}_n + \frac{\Delta t^2}{4}\tilde{\mathbf{a}}\right] = \tilde{\mathbf{f}} \tag{24}$$

with

$$\begin{aligned} \mathbf{q}_{n+1} &= \mathbf{q}_n + \Delta t\,\mathbf{v}_n + \frac{\Delta t}{2}\tilde{\mathbf{a}} \\ \mathbf{v}_{n+1} &= \mathbf{v}_n + \Delta t\,\tilde{\mathbf{a}} \end{aligned} \tag{25}$$

where

$$\tilde{\mathbf{a}} = \mathbf{a}_n + \frac{1}{1+\rho_\infty^{\mathrm{s}}}(\mathbf{a}_{n+1} - \mathbf{a}_n) \tag{26}$$

Note that we have

$$W_1 = \frac{1}{2} \;\; \textit{and} \;\; \phi = \frac{1-\rho_\infty^{\mathrm{s}}}{2(1+\rho_\infty^{\mathrm{s}})} \tag{27}$$

for V0:$\{(\rho_\infty^{\min}, \rho_\infty^{\max}, \rho_\infty^{\mathrm{s}}) = (1.0, 1.0, \rho_\infty^{\mathrm{s}})\}$ *with* ρ_∞^{s}. *We call U0/V0:* $\{(\rho_\infty^{\min}, \rho_\infty^{\max}, \rho_\infty^{\mathrm{s}}) = (1.0, 1.0, 1.0)\}$, *i.e.,* $\mathbf{a}_{n+1} \approx \ddot{\mathbf{q}}(t_{n+1})$, *and* *V0:*$\{(\rho_\infty^{\min}, \rho_\infty^{\max}, \rho_\infty^{\mathrm{s}}) = (1.0, 1.0, 0.0)\}$, *i.e.,* $\mathbf{a}_{n+1} \approx \ddot{\mathbf{q}}(t_{n+1/2})$, *the* ***midpoint rule with the endpoint acceleration (MRP-EPA)****, which is the classical midpoint rule, and the new* ***midpoint rule with the midpoint acceleration (MRP-MPA)****, respectively. Notice that the difference between MPR-EPA and MPR-MPA is*

$$\frac{\mathbf{v}_{n+1} - \mathbf{v}_n}{\Delta t} = \tilde{\mathbf{a}} = \begin{cases} \mathbf{a}_{n+1/2} \approx \frac{1}{2}[\ddot{\mathbf{q}}(t_{n+1}) + \ddot{\mathbf{q}}(t_n)] \;\; \textit{for MPR-EPA} \\ \mathbf{a}_{n+1} \approx \ddot{\mathbf{q}}(t_{n+1/2}) \;\; \textit{for MPR-MPA} \end{cases} \tag{28}$$

Note that the information regarding the acceleration at the previous time step is not required to evaluate the current acceleration, and consequently, the new MPR-MPA provides a a more robust computational algorithm for general dynamics applications in contrast to the classical MPR-EPA. In contrast to the single field form shown above, by eliminating the acceleration, Eq. (24)-*Eq.*(26) *can be readily reduced to the two-field form version of the classical symplectic midpoint rule which takes the form*

$$\begin{aligned} \mathbf{M}\frac{\mathbf{v}_{n+1}-\mathbf{v}_n}{\Delta t}+\mathbf{C}\frac{\mathbf{v}_{n+1}+\mathbf{v}_n}{2}+\mathbf{K}\frac{\mathbf{q}_{n+1}+\mathbf{q}_n}{2}&=\tilde{\mathbf{f}} \\ \frac{\mathbf{v}_{n+1}+\mathbf{v}_n}{2}&=\frac{\mathbf{q}_{n+1}-\mathbf{q}_n}{\Delta t} \end{aligned} \tag{29}$$

where $\tilde{\mathbf{f}}=\mathbf{f}(t_{n+1/2})$ *can be used for the algorithmic time-dependent external force instead of* $\tilde{\mathbf{f}}=(\mathbf{f}_n+\mathbf{f})/2$ *since* $\mathbf{f}(t_{n+\alpha})-\mathbf{f}_{n+\alpha}=\mathscr{O}(\Delta t^2)$ *for* $\alpha\in[0,1]$.

4. *In the derivation of the GSSSS family of algorithms, the conditions of the second-order time accuracy were imposed following the algorithms by design concept; see Zhou and Tamma (2004b). From the truncation error analysis for Eqs.* (1)-(5), *the necessary and sufficient conditions of the second-order time accuracy for the displacement and velocity are given as:*

$$\begin{aligned} &\lambda_1=\lambda_4=1, \\ &\lambda_2=\tfrac{1}{2},\ \lambda_5=\tfrac{1}{2}+W_1(\Lambda_6-\Lambda_4) \end{aligned} \tag{30}$$

and $\Lambda_1=\Lambda_4$ *for a homogeneous case* ($\mathbf{f}=\mathbf{0}$), *and* $\Lambda_1=\Lambda_4=1$ *for a non-homogeneous case* ($\mathbf{f}\neq\mathbf{0}$). *Notice that the above conditions (for the non-homogeneous case) are a priori for the U0 and V0 family-based algorithms. In addition to the conditions, Eq.* (30), *we must introduce* $\phi:=W_1(\Lambda_6-1)$ *to guarantee the second-order time accuracy not only of the displacement and velocity, but also of the acceleration. Note that we can never obtain the second order-time accuracy of the acceleration if we break the conditions, Eq.* (30), *and the orders of accuracy of the displacement and velocity are always dependent upon each other in this algorithmic framework. However, the second order-time accuracy of the acceleration is not the necessary condition for the second order-time accuracy of the displacement and velocity.*

Theorem 2 (Time Level Consistency)
In order to guarantee the second-order time accuracy in all time-dependent variables, such as $\mathbf{q}(t)$, $\dot{\mathbf{q}}(t)$, $\ddot{\mathbf{q}}(t)$, *and* $\mathbf{f}(t)$, *in the discrete equation of motion, they must be evaluated at the same time level* $t=t^*=t_{n+W_1}$ *as*

$$\mathbf{0}=\mathbf{M}\tilde{\mathbf{a}}+\mathbf{C}\tilde{\mathbf{v}}+\mathbf{K}\tilde{\mathbf{q}}-\tilde{\mathbf{f}}+\mathscr{O}(\Delta t^2)=\mathbf{M}\ddot{\mathbf{q}}(t^*)+\mathbf{C}\dot{\mathbf{q}}(t^*)+\mathbf{K}\mathbf{q}(t^*)-\mathbf{f}(t^*) \tag{31}$$

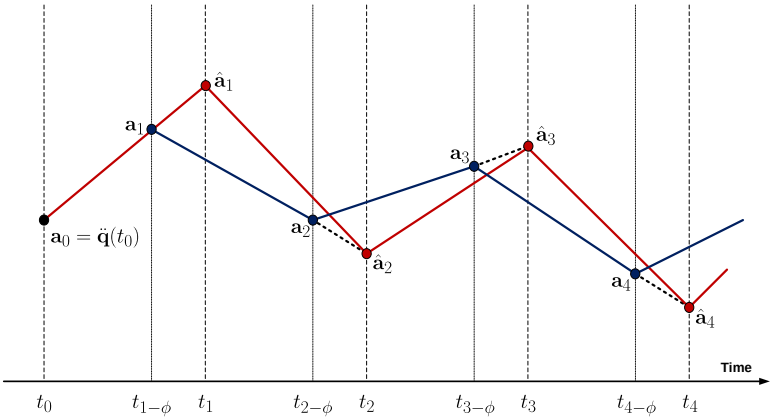


Figure 3: Illustration of the true acceleration history (red line).

Proof. By using the Taylor series expansions about time $t = t_n$, we get

$$\begin{aligned}\tilde{\mathbf{a}} &= (1 - W_1\Lambda_6)\ddot{\mathbf{q}}(t_{n-\phi}) + W_1\Lambda_6\ddot{\mathbf{q}}(t_{n+1-\phi}) + \mathscr{O}(\Delta t^2) \\ &= \ddot{\mathbf{q}}(t_n) + \Delta t W_1 \dddot{\mathbf{q}}(t_n) + \mathscr{O}(\Delta t^2) \\ &= \ddot{\mathbf{q}}(t_{n+W_1})\end{aligned} \tag{32}$$

when $\mathbf{a}_n - \ddot{\mathbf{q}}(t_{n-\phi}) = \mathscr{O}(\Delta t^2)$. Similarly, we get

$$\tilde{\mathbf{v}} = \dot{\mathbf{q}}(t_n) + \Delta t W_1 \ddot{\mathbf{q}}(t_n) + \mathscr{O}(\Delta t^2) = \dot{\mathbf{q}}(t_{n+W_1}) \tag{33}$$

$$\tilde{\mathbf{q}} = \mathbf{q}(t_n) + \Delta t W_1 \dot{\mathbf{q}}(t_n) + \mathscr{O}(\Delta t^2) = \mathbf{q}(t_{n+W_1}) \tag{34}$$

$$\tilde{\mathbf{f}} = \mathbf{f}(t_n) + \Delta t W_1 \dot{\mathbf{f}}(t_n) + \mathscr{O}(\Delta t^2) = \mathbf{f}(t_{n+W_1}) \tag{35}$$

Therefore, the discrete equation of motion is evaluated at the time level $t = t_{n+W_1}$ where the second-order time accuracy in $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$, and $\mathbf{f}$ are guaranteed. The time level $t^* = t_{n+W_1}$ is called the ***algorithmic time level***. If the algorithmic balance equation is satisfied at the time level $t^* = t_{n+W_1}$ with the error $\mathscr{O}(\Delta t^2)$, the acceleration, velocity, and configuration are of second-order time accuracy. The algorithmic time level consistency at time $t = t_{n+W_1}$ in the balance equation is illustrated in Fig. 4.

Remark 2.2 (Theorem 2)

1. ***Extension to nonlinear systems:*** *The time level consistency theorem plays a fundamental role when we extend and apply the GSSSS family of algorithms to nonlinear dynamical systems. The basic strategy is that we develop*

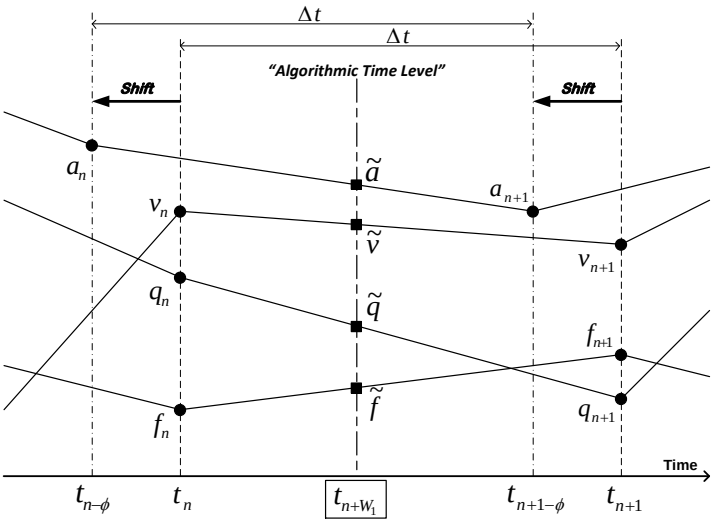


Figure 4: Illustration of the algorithmic time level

time integration schemes which satisfy the algorithmic time level consistency theorem at time level of t_{n+W_1} in order to guarantee the second order-time accuracy in all variables appear in discrete balance equations. See Shimada and Tamma (2012, 2013); Har and Tamma (2012) for detailed discussions.

2. ***First order systems:*** *The time level consistency theorem still holds for first-order systems generally represented as $\dot{\mathbf{q}} = \mathbf{F}(\mathbf{q},t)\ \forall t \in \mathbb{I}$ with given initial condition $\mathbf{q}_0 = \mathbf{q}(t_0)$. That is, we discretize the balance equation with the algorithms by design concept such that the time level consistency at time $t = t_{n+W_1}$ is guaranteed as follows:*

$$\mathbf{0} = \tilde{\mathbf{v}} - \mathbf{F}(\tilde{\mathbf{q}}, t_{n+W_1}) + \mathscr{O}(\Delta t^2) = \dot{\mathbf{q}}(t_{n+W_1}) - \mathbf{F}(\mathbf{q}(t_{n+W_1}), t_{n+W_1}) \tag{36}$$

where $\tilde{\mathbf{q}}$ and $\tilde{\mathbf{v}} = \mathbf{v}_{n+W_1\Lambda_6} := (1 - W_1\Lambda_6)\mathbf{v}_n + W_1\Lambda_6\mathbf{v}_{n+1}$ in the above equation denote the algorithmic $\mathbf{q}$ and $\mathbf{v}$ for the first-order system. The time level of $\mathbf{v}_n$ must satisfy $\mathbf{v}_n \approx \dot{\mathbf{q}}(t_{n-\phi})$ for $n \in \{1, 2, \cdots, \bar{n}\}$ with $\phi = W_1(\Lambda_6 - 1)$. The resulting algorithm is known as the GSSSS-1 (or GS4-1) family of algorithms; see Masuri, Sellier, Zhou, and Tamma (2011) for the detailed derivations and applications to linear transient first-order systems.

3 Implications of a Shifted Acceleration Time Level

U0 Family-based Algorithms: To gain a physical interpretation of acceleration time level shift denoted by ϕ, note that ϕ needs to be viewed separately for each of the U0 and V0 families of algorithms due to the fact that values of W_1 and Λ_6 are not the same. For the case of the family of U0 algorithms we have:

$$W_1 = \frac{1}{1+\rho_\infty^{\mathrm{s}}} \tag{37}$$

$$\Lambda_6 = \frac{2+\rho_\infty^{\min}+\rho_\infty^{\max}+\rho_\infty^{\mathrm{s}}-\rho_\infty^{\min}\rho_\infty^{\max}\rho_\infty^{\mathrm{s}}}{(1+\rho_\infty^{\min})(1+\rho_\infty^{\max})}; \tag{38}$$

Therefore, $\phi := W_1(\Lambda_6 - 1)$ can be written in terms of the roots as

$$\phi_{\mathrm{U0}} = \frac{1-\rho_\infty^{\max}\rho_\infty^{\min}}{(1+\rho_\infty^{\max})(1+\rho_\infty^{\min})} \tag{39}$$

Notice that ϕ_{U0} does not depend on ρ_∞^{s}. Recall that ϕ was defined to be the offset of acceleration from time level t_n. As such, the resulting values of acceleration for a given time step between t_n and t_{n+1} are actually at time $t_{n-\phi+1}$. Fig. 5a shows this time level vs. $\rho_\infty^{\min}$ and $\rho_\infty^{\max}$. We see that for the family of U0 algorithms the calculated accelerations occur at a time between t_n and t_{n+1}. In current practice it is being assumed by all researchers, that what all algorithms yield is the value of acceleration at time t_{n+1}; this results in a maximum possible error in time of Δt, as demonstrated later.

V0 Family-based Algorithms: In a similar fashion to the U0 family of algorithms, a physical interpretation of ϕ can also be shown by considering the associated possible range of values. For the V0 family of algorithms we have:

$$W_1 = \frac{3+\rho_\infty^{\min}+\rho_\infty^{\max}-\rho_\infty^{\min}\rho_\infty^{\max}}{2(1+\rho_\infty^{\min})(1+\rho_\infty^{\max})} \tag{40}$$

$$\Lambda_6 = \frac{2(2+\rho_\infty^{\min}+\rho_\infty^{\max}+\rho_\infty^{\mathrm{s}}-\rho_\infty^{\min}\rho_\infty^{\max}\rho_\infty^{\mathrm{s}})}{(1+\rho_\infty^{\mathrm{s}})(3+\rho_\infty^{\min}+\rho_\infty^{\max}-\rho_\infty^{\min}\rho_\infty^{\max})}; \tag{41}$$

Hence, we have:

$$\phi_{\mathrm{V0}} = \frac{1-\rho_\infty^{\mathrm{s}}}{2(1+\rho_\infty^{\mathrm{s}})} \tag{42}$$

Notice that ϕ_{V0} depends on ρ_∞^{s} only. From Eq. 42 we see that it is possible to describe and construct a plot of acceleration time level vs. ρ_∞^{s} (see Fig. 5b). Notice

that in the case of V0, the acceleration resulting from a single time step between t_n and t_{n+1} is located between $t_{n+\frac{1}{2}}$ and t_{n+1} resulting in a maximum possible error in time of $\frac{\Delta t}{2}$, as demonstrated later.

Optimal Algorithms: From Eq. (39) or Eq. (42), we can obtain ϕ for the U0V0/V0U0 optimal family of algorithms, i.e., U0V0/V0U0($\rho_\infty, 1, \rho_\infty$) where $\rho_\infty := \rho_\infty^{\min} = \rho_\infty^{\mathrm{s}}$, as[2]

$$\phi_{\mathrm{U0V0/V0U0_{opt}}} = \frac{1-\rho_\infty}{2(1+\rho_\infty)} \tag{43}$$

Notice that ϕ for the V0U1 optimal family of algorithms, i.e., algorithms given by setting $\rho_\infty := \rho_\infty^{\min} = \rho_\infty^{\max} = \rho_\infty^{\mathrm{s}}$ in the V0 family of algorithms, has the same ϕ as shown in Eq. (44). On the other hand, ϕ for the U0V1 optimal family of algorithms, i.e., algorithms given by setting $\rho_\infty := \rho_\infty^{\min} = \rho_\infty^{\max} = \rho_\infty^{\mathrm{s}}$ in the U0 family of algorithms, which is identical to the Generalized-α method, has the following ϕ:

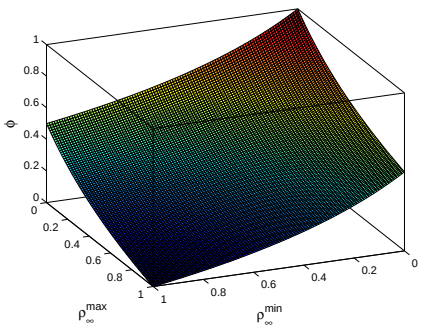
$$\phi_{\mathrm{U0V1_{opt}}} = \frac{1-\rho_\infty^2}{(1+\rho_\infty)^2} \tag{44}$$

These ϕ in terms of $\rho_\infty \in [0,1]$ are shown in Fig. 5c.

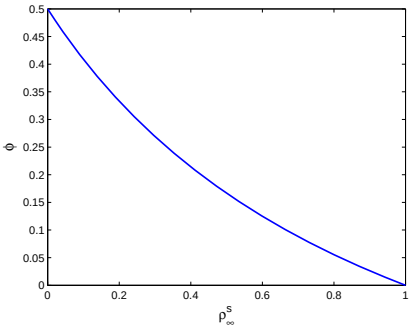
Note that only U0V0($1, 1, \rho_\infty^{\mathrm{s}}$) family of algorithms yields $\phi = 0$ within the implicit GSSSS family of algorithms. Also note that numerically dissipative *implicit* schemes in either U0 or V0 families possess $\phi \neq 0$ in general. For the explicit GSSSS family of algorithms ($\eta_1 = \eta_2 = 0$), V0($\rho_\infty^{\min}, \rho_\infty^{\max}, 1$) family of algorithms as well as U0V0($1, 1, \rho_\infty^{\mathrm{s}}$) family of algorithms yield $\phi = 0$ since the relation given in Eq. (6) is no longer required to be satisfied for explicit schemes.

To fully understand the impact of the above, consider the hypothetical situation of running a simulation to an end time of 0.5 sec (assume $t_0 = 0$) with a time step size $\Delta t = 0.1$ sec with a numerically dissipative algorithm U0V0($0, 1, 0$)≡U0($0, 1, 0$) which yields $\phi = 0.5$. The output of this hypothetical simulation would result in values of displacement, velocity, and acceleration at times seen in the following table.

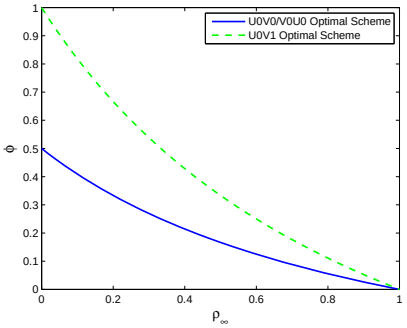
[2] Algorithms within the U0 family and V0 family are generally expressed as U0($\rho_\infty^{\min}$,$\rho_\infty^{\max}$,ρ_∞^{s}) and V0($\rho_\infty^{\min}$,$\rho_\infty^{\max}$,ρ_∞^{s}), respectively.
If $\rho_\infty^{\max} = 1$, any algorithms within the GSSSS family possess zero-th order overshoots in both displacement and velocity; therefore, U0($\rho_\infty^{\min}$,1,ρ_∞^{s}) ≡ U0V0($\rho_\infty^{\min}$,1,ρ_∞^{s}) and V0($\rho_\infty^{\min}$,1,ρ_∞^{s}) ≡ V0U0($\rho_\infty^{\min}$,1,ρ_∞^{s}).
If $\rho_\infty^{\max} \neq 1$, the U0 and V0 family of algorithms possess first-order overshoots in the velocity and displacement, respectively; that is, U0($\rho_\infty^{\min}, \rho_\infty^{\max}, \rho_\infty^{\mathrm{s}}$) ≡ U0V1($\rho_\infty^{\min}, \rho_\infty^{\max}, \rho_\infty^{\mathrm{s}}$) and V0($\rho_\infty^{\min}, \rho_\infty^{\max}, \rho_\infty^{\mathrm{s}}$) ≡ V0U1($\rho_\infty^{\min}, \rho_\infty^{\max}, \rho_\infty^{\mathrm{s}}$).



(a) U0 Family



(b) V0 Family



(c) U0V0/V0U0 and U0V1 Optimal Families

Figure 5: Time level of the resulting acceleration over time step t_n to t_{n+1}

n	Time [sec]	Displacement	Velocity	Acceleration	True Acceleration
0	0.0	$\mathbf{q}_0 = \mathbf{q}(0.0)$	$\mathbf{v}_0 = \dot{\mathbf{q}}(0.0)$	$\mathbf{a}_0 = \ddot{\mathbf{q}}(0.00)$	$\hat{\mathbf{a}}_0 = \ddot{\mathbf{q}}(0.0)$
1	0.1	$\mathbf{q}_1 \approx \mathbf{q}(0.1)$	$\mathbf{v}_1 \approx \dot{\mathbf{q}}(0.1)$	$\mathbf{a}_1 \approx \ddot{\mathbf{q}}(0.05)$	$\hat{\mathbf{a}}_1 = 2\mathbf{a}_1 - \mathbf{a}_0 \approx \ddot{\mathbf{q}}(0.1)$
2	0.2	$\mathbf{q}_2 \approx \mathbf{q}(0.2)$	$\mathbf{v}_2 \approx \dot{\mathbf{q}}(0.2)$	$\mathbf{a}_2 \approx \ddot{\mathbf{q}}(0.15)$	$\hat{\mathbf{a}}_2 = 1.5\mathbf{a}_2 - 0.5\mathbf{a}_1 \approx \ddot{\mathbf{q}}(0.2)$
3	0.3	$\mathbf{q}_3 \approx \mathbf{q}(0.3)$	$\mathbf{v}_3 \approx \dot{\mathbf{q}}(0.3)$	$\mathbf{a}_3 \approx \ddot{\mathbf{q}}(0.25)$	$\hat{\mathbf{a}}_3 = 1.5\mathbf{a}_3 - 0.5\mathbf{a}_2 \approx \ddot{\mathbf{q}}(0.3)$
4	0.4	$\mathbf{q}_4 \approx \mathbf{q}(0.4)$	$\mathbf{v}_4 \approx \dot{\mathbf{q}}(0.4)$	$\mathbf{a}_4 \approx \ddot{\mathbf{q}}(0.35)$	$\hat{\mathbf{a}}_4 = 1.5\mathbf{a}_4 - 0.5\mathbf{a}_3 \approx \ddot{\mathbf{q}}(0.4)$
5	0.5	$\mathbf{q}_5 \approx \mathbf{q}(0.5)$	$\mathbf{v}_5 \approx \dot{\mathbf{q}}(0.5)$	$\mathbf{a}_5 \approx \ddot{\mathbf{q}}(0.45)$	$\hat{\mathbf{a}}_5 = 1.5\mathbf{a}_5 - 0.5\mathbf{a}_4 \approx \ddot{\mathbf{q}}(0.5)$

An outcome of this shift in the acceleration time level is that the notation implied (naively) in the selected algorithm leads to an interpretation that is somewhat misleading. The algorithm in the a-form solves for $\Delta\mathbf{a} = \mathbf{a}_{n+1} - \mathbf{a}_n \approx \ddot{\mathbf{q}}(t_{n+1-\phi}) - \ddot{\mathbf{q}}(t_{n-\phi})$, and then uses the updates to obtain values of $\mathbf{v}_{n+1} \approx \dot{\mathbf{q}}(t_{n+1})$ and $\mathbf{q}_{n+1} \approx \mathbf{q}(t_{n+1})$. As a result, what would be considered the output/results are actually values of $\mathbf{q}_{n+1}$ and $\mathbf{v}_{n+1}$ at time $t = t_{n+1} = t_0 + (n+1)\Delta t$ over the duration of the

run-time and values of $\mathbf{a}$ at time $t = t_{n+1-\phi} = t_0 + (n+1-\phi)\Delta t$, even though over the duration of the run-time the equation of motion was satisfied consistently at the same time level. As such, any algorithm that does not satisfy $\phi = 0$ produces data that is very easily misrepresented. The current understanding before the findings of this paper was that all time integration algorithms simply returned values of $\mathbf{q}_{n+1}$, $\mathbf{v}_{n+1}$, and $\mathbf{a}_{n+1}$ at time t_{n+1}. This is strictly not not the case, in general. Therefore, the correct time history of $\mathbf{q}_{n+1}$, $\mathbf{v}_{n+1}$, and $\mathbf{a}_{n+1}$ should in fact appear with accelerations misaligned with displacements and velocities. To reiterate, if the goal of the hypothetical simulation described above was to obtain numerical values of $\mathbf{q}$, $\dot{\mathbf{q}}$, and $\ddot{\mathbf{q}}$ at time $t = 0.5$ sec, in general any of the algorithms which yield $\phi \neq 0$ will *not* provide all three quantities correctly. This concept has extremely important implications about the manner in which numerically dissipative algorithms, as well as some non-dissipative methods, are used with respect to accelerations for the class of LMS methods.

4 Interpretation and Description of a Consistent Convergence Plot

Following the discussion and illustration of the algorithmic time level concept, we now are able to address the fact that in general, for numerically dissipative algorithms, to-date one is not able to demonstrate that the accelerations are second order time accurate but do obtain second order time accuracy for both displacements and velocities. For numerically non-dissipative algorithms such as the velocity based scheme, this also happens to be the case. To understand the resulting consequences of the shift in the time level which the algorithm returns on a convergence plot, we first must focus on how the plot is produced.

To construct a converge plot (notice here that convergence is always meant to mean convergence in time, not convergence of the spatial discretization), the same problem is run multiple times with only a change in the time step size. The time step is gradually reduced and the slope of the result yields the order of time accuracy of the variable. An end time of the simulation is chosen and the resulting displacement, velocity, and acceleration are compared to values of the exact solution at that time. If, as in most practical problems, there is no exact known analytic solution, then the simulation is run with a time step which is very small in comparison to the others and this run is considered to be the benchmark solution. The log of the error in the numerical solution is then plotted versus the log of the time step size used in the solution. The slope of the line generated by these points is then the convergence rate of the algorithm.

To visualize how the final step of the simulations used to generate the acceleration convergence plot looks like, Figs. 6 are provided for illustration. The assumption is the simulation will be run using several different time step sizes, for example:

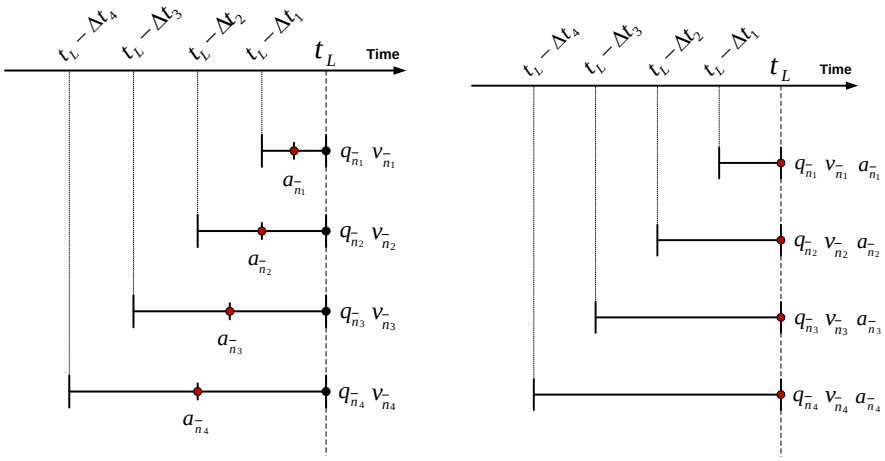
$\Delta t_1 = \Delta t$, $\Delta t_2 = 2\Delta t$, $\Delta t_3 = 3\Delta t$, and $\Delta t_4 = 4\Delta t$. Without properly accounting for the acceleration offset, the results will show second order time accuracy for all three quantities only if $\phi = 0$, and otherwise will show second order time accuracy for the displacement and velocity, and only first order time accuracy in the acceleration as reported in the literature Erlicher, Bonaventura, and Bursi (2002). If the time step sizes are chosen as to properly align accelerations, then the comparison of the numerical solutions to the exact solution are occurring at the same time level. It is then, that these normally misrepresented accelerations can be readily proven to be second order time accurate.

To properly describe the convergence plot in which the value of acceleration from the numerical simulations are compared with the acceleration of the exact solution at precisely the same time, very close attention must be paid to the selection of the time step size for each simulation used in generating the plot. As an example, assume that the convergence plot will have 4 data points meaning that one has 4 numerical solutions which are being compared to an exact solution. Instead of defining 4 different time step sizes, let us define the total number of steps for each of the 4 solutions that we will use. Then calculate the corresponding time step size as follows:

$$\Delta t_i = \frac{T}{\bar{n}_i + \phi} = \frac{t_L - t_0}{\bar{n}_i + \phi} \tag{45}$$

where $\bar{n}_i$ denotes the total number of time steps with correspondence with Δt_i (the acceleration convergence rate with Δt_i is plotted at time t_L).

It is shown below in the different numerical examples that using this procedure in fact results in properly obtaining second order accurate accelerations, thereby validating the assumption of the offset in the acceleration time level. The first example problem is a simple linear dynamic single degree of freedom problem for which we have an exact solution. The next example is a multi-DOF dynamic problem with nonlinear strain definition (Green strain). For illustration, a general purpose dynamic research code was developed to simulate any combination of springs and masses. Having tested the current assertions on codes ranging from single DOF to multi-DOF and both linear and non-linear dynamic problems, we have great confidence that the current work will advance the field of general numerically dissipative (and numerically non-dissipative as well) LMS class of algorithms. This will be of benefit in future ither developments and advances.



(a) $\phi = 0.5$ (b) $\phi = 0.0$

Figure 6: Examples of the final time steps used in creating a convergence plot when (a) $\phi = 0.5$ and (b) $\phi = 0.0$. For (a), the acceleration time level shift is required.

5 Numerical Illustrations

5.1 Single Degree of Freedom (SDOF) Example

Consider a single degree of freedom problem, $\ddot{q}(t) + 0.25\dot{q}(t) + 10q(t) = 0$ with given initial conditions $q_0 = q(0) = 2$ and $v_0 = \dot{q}(0) = 2$. For this system it was shown in Fig. 1c for the V0U0(1,1,0) algorithm (MPR-MPA algorithm), the acceleration appears to be only first order accurate. Using the concepts described above, we are now able to understand why: the ϕ value for the V0U0(1,1,0) algorithm is 0.5. This means that the accelerations resulting from the algorithm do not align with the displacement and velocity, and care must be taken to properly represent the converge rate. Fig. 7 shows the resulting convergence plot taking into account this shift; notice that now all three primary variables show the expected order of time accuracy.

To demonstrate that the above procedure indeed yields expected second order time accurate accelerations over the entire range of ϕ, four common algorithms which which include numerical dissipation in the U0 family of algorithms were selected as shown in Table 2.

For each of the four algorithms described above, convergence plots are shown using 4 numerical solutions. Two convergence plots were generated for each algorithm:

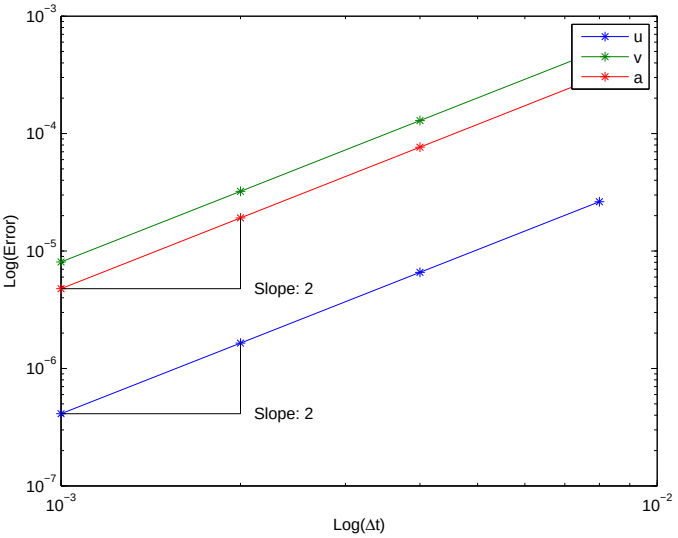


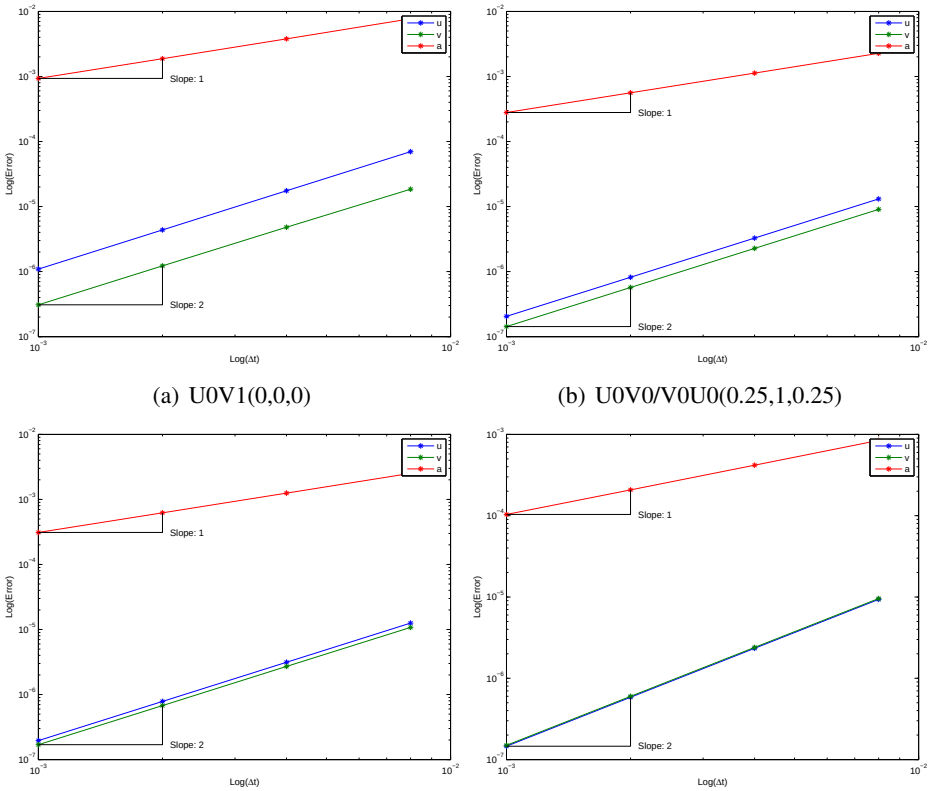
Figure 7: V0U0(1,1,0): MPR-MPA method - acceleration aligned convergence plot (see improvement and compare to Fig. 1c)

Table 1: Selected algorithms from the U0 family for Figs. 8 and 9 and Figs. 11 and 12

Algorithm Family($\rho_\infty^{\min}$,$\rho_\infty^{\max}$,ρ_∞^{s})	Common family name	Acceleration time level
U0V1(0,0,0)	$\in$ WBZ	$\mathbf{a}_{n+1} \approx \ddot{\mathbf{q}}(t_n)$
U0V0/V0U0(0.25,1,0.25)	$\in$ U0V0/V0U0-optimal	$\mathbf{a}_{n+1} \approx \ddot{\mathbf{q}}(t_{n+0.7})$
U0V1(0.5,0.5,0.5)	$\in$ U0V1-optimal	$\mathbf{a}_{n+1} \approx \ddot{\mathbf{q}}(t_{n+0.6667})$
U0V1(0.8,0.8,0.125)	$\in$ HHT-α	$\mathbf{a}_{n+1} \approx \ddot{\mathbf{q}}(t_{n+0.8889})$

Table 2: Selected algorithms from the V0 family for Figs. 13 and 14

Algorithm Family($\rho_\infty^{\min}$,$\rho_\infty^{\max}$,ρ_∞^{s})	Family name	Acceleration time level
V0U1(0,0,0)	$\in$ *V0-based* WBZ	$\mathbf{a}_{n+1} \approx \ddot{\mathbf{q}}(t_{n+0.5})$
V0U0/U0V0(0.25,1,0.25)	$\in$ V0U0/U0V0-optimal	$\mathbf{a}_{n+1} \approx \ddot{\mathbf{q}}(t_{n+0.7})$
V0U1(0.5,0.5,0.5)	$\in$ V0U1-optimal	$\mathbf{a}_{n+1} \approx \ddot{\mathbf{q}}(t_{n+0.8333})$
V0U1(0.8,0.8,0.125)	$\in$ *V0-based* HHT-α	$\mathbf{a}_{n+1} \approx \ddot{\mathbf{q}}(t_{n+0.6111})$



(a) U0V1(0,0,0) (b) U0V0/V0U0(0.25,1,0.25)

(c) U0V1(0.5,0.5,0.5) (d) U0V1(0.8,0.8,0.125)

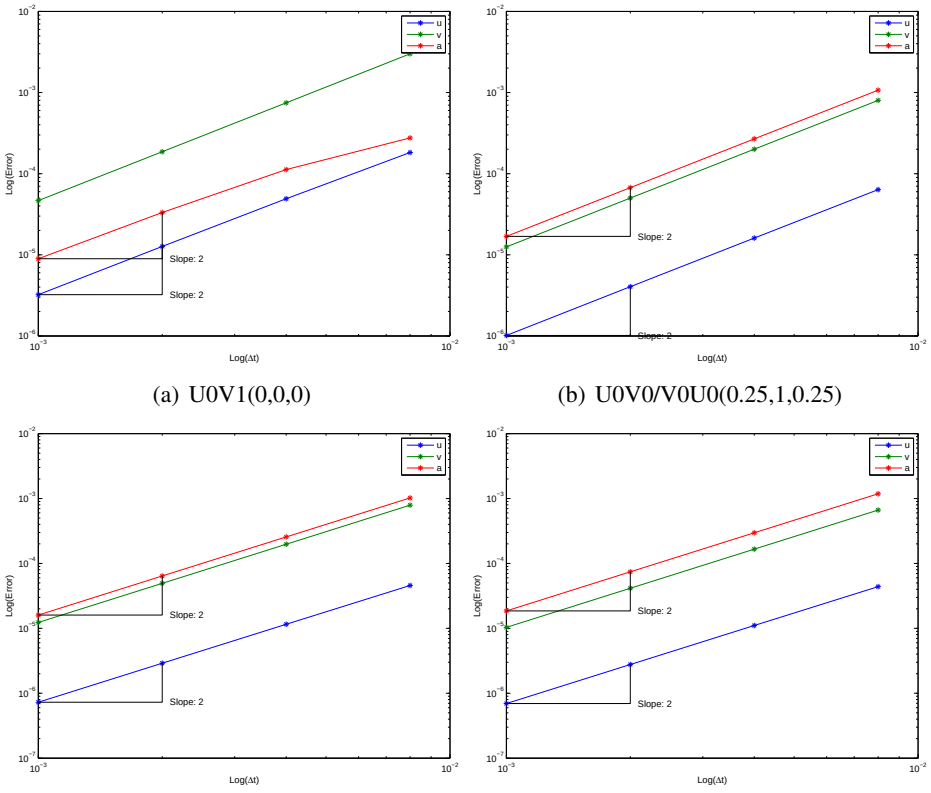
Figure 8: SDOF U0 family traditional convergence plots showing poor understanding

one without accounting for acceleration offset and one in which accelerations were aligned via Eq. 45.

It is consistently shown in Figs. 8 and 9 that in the traditional convergence plot the slopes of the displacement and velocity are 2.0 while the slope of the acceleration is 1.0. Once aligned correctly, the proper convergence plots now indeed show the acceleration has a precise slope of 2.0.

5.2 Nonlinear Tetrahedral Spring-mass System Example

Consider another example, which is a tetrahedral spring-mass system depicted in Fig. 10a constructed of six springs and four masses. Each spring has initial length of 1 m, stiffness of 10^3 N/m, and mass 1 kg. The spring is charac-



(a) U0V1(0,0,0)

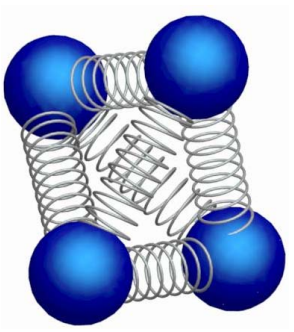
(b) U0V0/V0U0(0.25,1,0.25)

(c) U0V1(0.5,0.5,0.5)

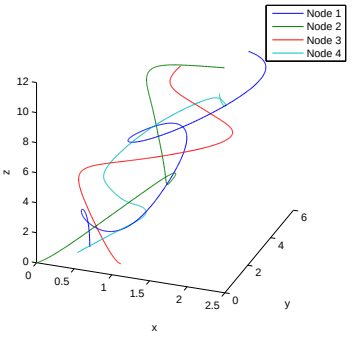
(d) U0V1(0.8,0.8,0.125)

Figure 9: SDOF U0 family acceleration aligned convergence plots showing improved understanding of fundamentals

terized by geometrically nonlinear strain (Green strain). The initial position of the four nodes are $[0.5, 3^{0.5}/2, 0, 0, 0, 0, 1, 0, 0, 0.5, 1/(2*3^{0.5}), (2/3)^{0.5}]^T$ [m] with given initial displacement of $[0, 0.5, 0.2, 0, 0, 0, 0, 0, 0.8, 0, 0, 0, 0]^T$ [m] and initial velocity of $[0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 1, 3, 2]^T$ [m/s]. The dynamic responses of the system over a 5 sec time span with a time step size $\Delta t = 0.1$ sec can be seen in Fig. 10b. Figs. 11 and 12 show the misaligned and aligned results that accurately describe the correct evaluation of accelerations, using the selected algorithms from the U0 family as shown in Table 1. Similarly, Figs. 13 and 14 show the misaligned and aligned results that accurately describe the correct evaluation of accelerations, using the selected algorithms from the V0 family as shown in Table 2. Note that U0V0 optimal schemes and V0U0 optimal schemes are identical each other.

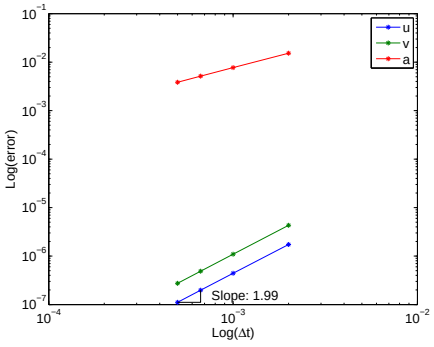


(a) Geometry

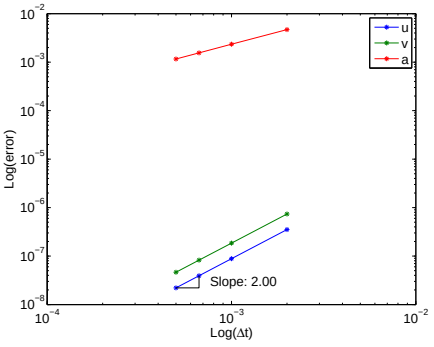


(b) Dynamic Response

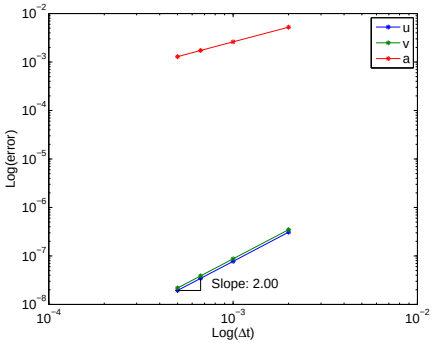
Figure 10: Tetrahedral spring-mass problem



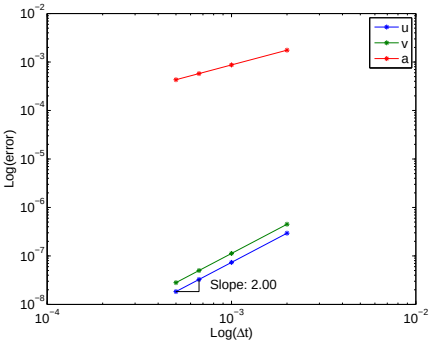
(a) U0V1(0,0,0)



(b) U0V0(0.25,1,0.25)

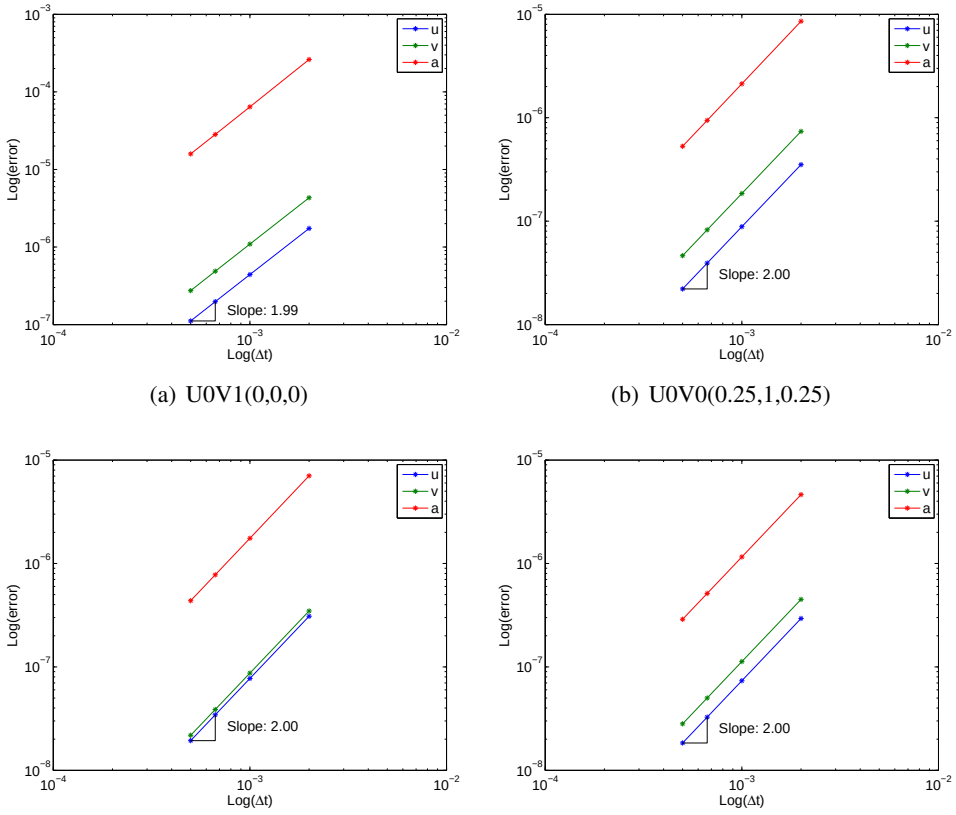


(c) U0V1(0.5,0.5,0.5)



(d) U0V1(0.8,0.8,0.125)

Figure 11: Tetrahedral spring-mass U0 family traditional convergence plots showing poor understanding



(a) U0V1(0,0,0)

(b) U0V0(0.25,1,0.25)

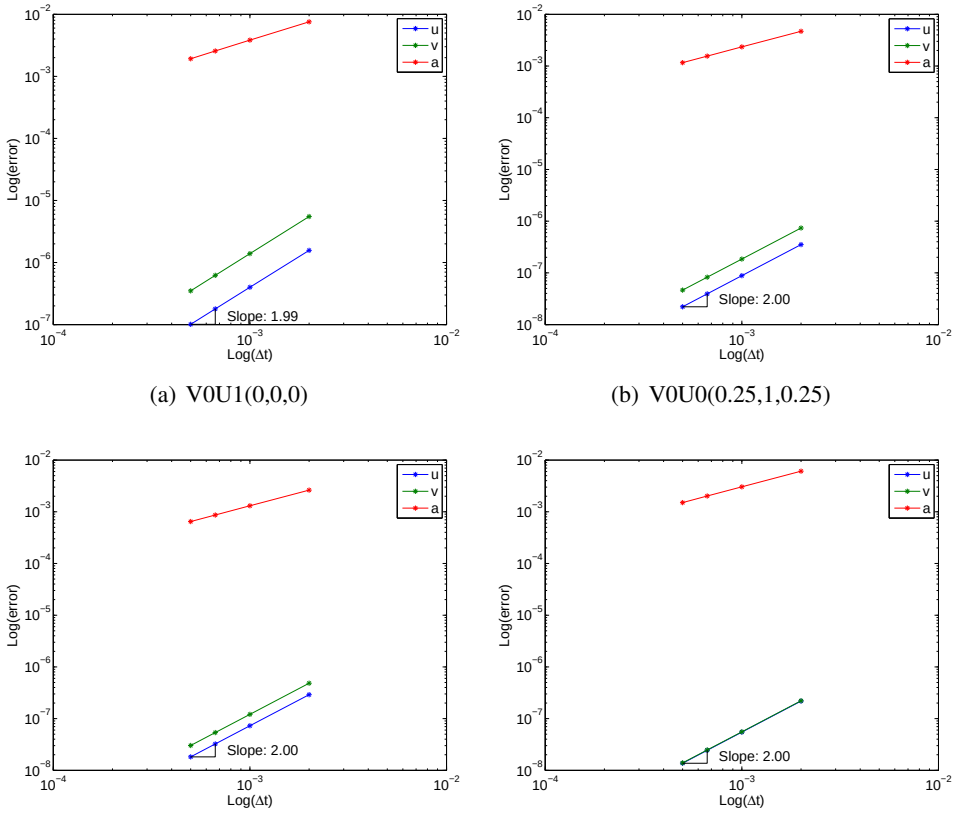
(c) U0V1(0.5,0.5,0.5)

(d) U0V1(0.8,0.8,0.125)

Figure 12: Tetrahedral spring-mass U0 family acceleration aligned convergence plots showing improved understanding of fundamentals

6 Conclusion

This paper described the accurate and precise evaluation of acceleration computations useful for structural dynamic simulations. The focus was upon the class of LMS methods which are predominantly used in most commercial and research software. They include numerically non-dissipative and numerically dissipative methods. An underlying theory regarding the time levels at which these computations need to take place precisely to achieve the desired order of time accuracy which is targeted at second order for the class of LMS methods was presented. The literature to-date lacks a theoretical basis on how to evaluate these acceleration computations accurately and precisely. The LMS methods have been developed under the umbrella of GSSSS algorithms which encompass most of the developments to-date



(a) V0U1(0,0,0)

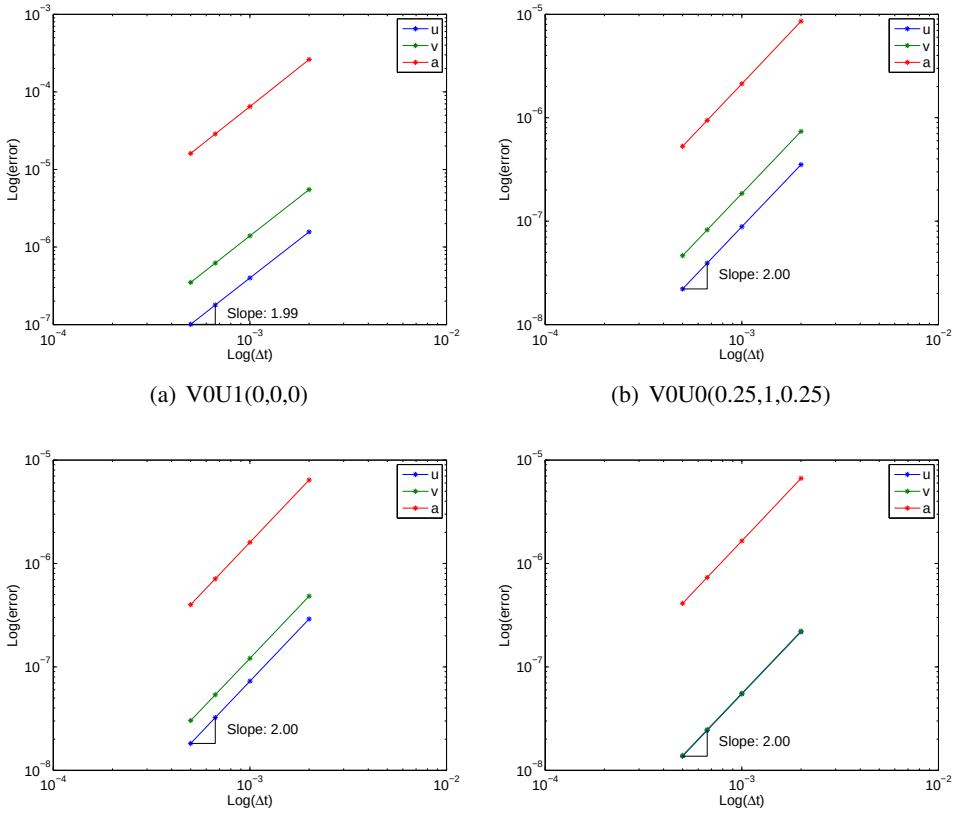
(b) V0U0(0.25,1,0.25)

(c) V0U1(0.5,0.5,0.5)

(d) V0U1(0.8,0.8,0.125)

Figure 13: Tetrahedral spring-mass V0 family traditional convergence plots showing poor understanding

and contain both numerically non-dissipative and numerically dissipative methods. The fundamental problem is that with the exception of the standard Newmark and midpoint rule, the acceleration computations of all the others are not trivial and the resulting computations to achieve second order accuracy are not strictly at time level $n+1$. In this regard, we described the subtle issues and some noteworthy perspectives to accurately obtain the acceleration computations for general algorithms and the precise time levels that underlie their basic developments. We have provided a theoretical basis and an in-depth understanding to estimate accurately the accelerations and to ensure that they are all second order time accurate. Several simple numerical examples demonstrated the basic ideas.



(a) V0U1(0,0,0)

(b) V0U0(0.25,1,0.25)

(c) V0U1(0.5,0.5,0.5)

(d) V0U1(0.8,0.8,0.125)

Figure 14: Tetrahedral spring-mass V0 family acceleration aligned convergence plots showing improved understanding of fundamentals

Acknowledgement: Other related support in form of computer grants from the Minnesota Supercomputer Institute (MSI), Minneapolis, Minnesota is also gratefully acknowledged.

References

Erlicher, S.; Bonaventura, L.; Bursi, O. S. (2002): The Analysis of the Generalized-α Method for Nonlinear Dynamic Problems. *Computational Mechanics*, vol. 28, pp. 83–104.

Har, J.; Tamma, K. K. (2012): *Advances in Computational Dynamics of Particles, Materials and Structures*. John Wiley, Chichester,UK.

Hilber, H. M.; Hughes, T. J. R.; Taylor, R. L. (1977): Improved Numerical Dissipation for Time Integration Algorithms in Structural Dynamics. *Earthquake Engineering and Structural Dynamics*, vol. 5, pp. 283–292.

Masuri, S. U.; Sellier, M.; Zhou, X.; Tamma, K. K. (2011): Design of Order-preserving Algorithms for Transient First-order Systems with Controllable Numerical Dissipation. *Int. J. Num. Methods in Engr.*, vol. 88, pp. 1411–1448.

Newmark, N. M. (1959): A Method of Computation for Structural Dynamics. *Journal for American Society of Civil Engineers*, vol. 1, pp. 67–94.

Shimada, M.; Tamma, K. K. (2012): Conserving/Dissipative Algorithms and Designs for a System of N Particles: Total Energy Framework and Single-Field Form. *Computers and Structures*, vol. 112-113, pp. 380–405.

Shimada, M.; Tamma, K. K. (2013): Implicit Time Integrators and Designs for Nonlinear Second-Order Transient Systems: Elastodynamics. *Encyclopedia of Thermal Stresses*, vol. 5, pp. 2409–2416.

Tamma, K. K.; Kanapady, R.; Zhou, X.; Sha, D. (2000): Recent Advances in Computational Structural Dynamics Algorithms. In *Seventh Int-Conf. on Recent Advances in Structural Dynamics ISVR*, Southampton, UK.

Tamma, K. K.; Namburu, R. R. (1990): Applicability and Evaluation of An Implicit Self-Starting Unconditionally Stable Methodology for Dynamics of Structures. *Computers and Structures*, vol. 34, pp. 835–842.

Tamma, K. K.; Zhou, X.; Sha, D. (2000): The Time Dimension: A Theory of Development/Evolution, Classification, Characterization and Design of Computational Algorithms for Transient/Dynamic Applications. *Archives in Computational Mechanics*, vol. 7, no. 2, pp. 67–290.

Zhou, X.; Tamma, K. K. (2004): A New Unified Theory Underlying Time Dependent Linear First-Order Systems: A Prelude to Algorithms by Design. *International Journal for Numerical Methods in Engineering*, vol. 60, pp. 1699–1740.

Zhou, X.; Tamma, K. K. (2004): Design, Analysis, and Synthesis of Generalized Single Step Single Solve and Optimal Algorithms for Structural Dynamics. *International Journal for Numerical Methods in Engineering*, vol. 59, pp. 597–668.

Zhou, X.; Tamma, K. K. (2006): Algorithms by Design with Illustrations to Solid and Structural Mechanics/Dynamics. *International Journal for Numerical Methods in Engineering*, vol. 66, pp. 1738–1790.