HIGH-ORDER MULTISTEP ASYNCHRONOUS SPLITTING METHODS (MASM) FOR THE NUMERICAL SOLUTION OF MULTIPLE-TIME-SCALE ORDINARY DIFFERENTIAL EQUATIONS

BY

PRADEEP KUMAR GUDLA

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Mechanical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2016

Urbana, Illinois

Doctoral Committee:

Associate Professor Matthew West, Chair
Professor Geir Dullerud
Professor Alexander Vakakis
Associate Professor Srinivasa Salapaka
Assistant Professor Andreas Kloeckner

Abstract

Often one encounters dynamical systems containing a wide range of natural frequencies. These *multiple-time-scale systems*, which are modeled using stiff ordinary differential equations, are well known to present a significant challenge in obtaining a numerical solution efficiently. Multiple-time-scale systems can be broadly classified into two classes: (1) systems with well-separate discrete time scales, such as molecular dynamic simulations and electrical networks, and (2) systems with a continuously-distributed range of time scales, such as aerosol dynamics, multiscale structural dynamics and turbulent fluid flow. For the numerical simulation of systems with well-separated discrete time scales one can frequently average over the fast time scales, either analytically or numerically. This results in effective models with only slower time scales and allows efficient numerical simulations with large timesteps. In cases where this is not possible, either due to system complexity or the fact that there is simply a wide range of timescales with no clear scale separation, such as the continuously-distributed time scales systems, it has traditionally been necessary to simulate the entire system at the rate of the fastest timescale, which can be very expensive.

To efficiently simulate multiple-time-scale systems, many researchers have developed multiple-time-step numerical integration methods, where more than one timestep are used. These advance different components of the system forward in time at different rates, so that faster components can use small timesteps, while slower components can use large timesteps, resulting in lower computational cost. Most multiple-time-step integrators only apply to systems with discrete time scales, where subcycling methods, mollified methods, and r-RESPA are good examples. In addition, these methods which have several numerical timesteps re-

quire that timestep ratios be integer multiples of each other. In contrast, one family of multiple-time-step methods does not attempt to enforce any such restrictions, namely asynchronous integrators. These methods have incommensurate timesteps, such that all system components never synchronize at common time instants. This feature allows some asynchronous methods to be efficiently applied to systems with continuously-distributed time scales, where every time scale can have an appropriately-chosen numeral timestep. However, currently known asynchronous methods are at most second-order accurate and are known to suffer from resonance instabilities, severely limiting their practical efficiency gain relative to their synchronous counterparts.

In the present work, a new family of high-order Multistep Asynchronous Splitting Methods (MASM) is developed, based on a generalization of both classical linear multistep methods and the previously-known Asynchronous Splitting Methods (ASMs). These new methods compute high-order trajectory approximations using the history of system states and force vectors as for linear multistep methods, while at the same time allowing incommensurate timesteps to be used for different system components as in ASMs. This allows them to be both high-order and asynchronous, and means that they are applicable to systems with either discrete time scales or continuously-distributed time scales.

Consistency and convergence are established for these new high-order asynchronous methods both theoretically and via numerical simulations. For convergence, the only requirement is that the ratio of smallest to largest timestep remains bounded from above as the timesteps tend to zero. For a sufficiently regular ODE systems, an m-step MASM can achieve m^{th} order accuracy, which is proven analytically and then validated using numerical experiments. Numerical simulations show that these methods can be substantially more efficient than their synchronous counterparts. Given that appropriate timesteps are chosen, the efficiency gain using MASM compared to synchronous multi-step methods largely depends upon the force field splitting used.

MASM is proven to be a stable method, provided it is convergent. The stability criterion

also strongly depends upon the splitting of the force field chosen. In case of linear systems for which the Jacobian of the force vector is diagonalizable, the force vector splitting can be classified into *asynchronous splitting*, where each eigen-component is lumped with one of the component force vector, and *time scale splitting*, where an eigen-component is split between two or more component force vectors. Any force vector splitting is in general a combination of asynchronous splitting and time scale splitting, where some eigen-components are lumped with one of the component force vectors and other eigen-components are split between different component force vectors. For synchronous splitting we prove a stability condition with a bound essentially the same as for the corresponding synchronous multi-step method, while for time scale splitting we restrict the analysis to two-component systems and derive stability conditions for both conservative and non-conservative systems.

Finally, we also present an efficient time step selection (TSS) strategy that can be employed while using MASM for numerically solving ODEs, yielding the TSS-MASM method. This time step selection strategy is based on an optimal equidistribution principle, where component timesteps are chosen so that the contribution from each split force field towards the local discretization error is equal. The efficiency gain is system dependent and splitting dependent and is investigated numerically. For strongly coupled systems such as a multi-scale spring mass damper, TSS-MASM has approximately the same efficiency as synchronous multistep methods, while for weakly coupled systems such as aerosol condensation, TSS-MASM is much more efficient.

To my Dad, who paid the ultimate price for this work. I'm sure he would be very proud, if he had the chance to see it.

Table of Contents

List of Abbreviations

ODE	Ordinary Differential Equations.
MASM	Multistep based Asynchronous Splitting Methods.
ASM	Asynchronous Splitting Methods.
AVI	Asynchronous Variational Integrators
ABM	*Adams-Bashforth* Methods.
MTS	Multiple-Time Step methods.
TSS	Time Step Selection

List of Symbols

$\mathbf{X}(t)$	State vector
$\mathbf{F}$	Force vector
$\mathbf{F}^{[j]}$	j^{th} component force vector
$\mathbf{\Phi}$	Numerical state vector
Δt_k	Simulation timestep at time t_k
t_f	Final time
$t_k^{[j]}$	Time grid point on j^{th} component time grid (See Eq. 2.6)
$\Delta^{[j]}$	j^{th} component time grid (See Eq. 2.9)
Δ	Global time grid (See Eq. 2.10)
Δt_{asynch}	*Maximum asynchronicity* (See Eq. 2.8)
$\boldsymbol{\beta}^{[j]}$	Vector constant
$\tilde{\boldsymbol{\beta}}^{[j]}$	Vector constant
$\hat{\boldsymbol{\beta}}^{[j]}$	Vector constant
$\Psi_{\Delta t}$	One-step numerical integrator
N_f	Number of split force vectors
N	Dimensionality of state vector
M_k	Scalar constant (See Eq 2.14)
M	Upper bound to M_k
$\ell^{[j]}(*)$	Index mapping function mapping indices from global grid to j^{th} component time grid
$\mathbf{V}_k^{[*]}$	*Vandermonde matrix* evaluated at time t_k (See Eq. 2.20)

$\mathbf{R}$	Vector constant
$\tau_*^{[*]}$	Scalar constant (See Eq. 2.20)
τ	User specified *error tolerance*
L	*Lipschitz* constant
Λ	Increment function (See Eq. 2.39)
γ	Cost ratio (See Eq. 2.78)
$\mathbf{A}$	Constant matrix
h_{max}	Upper bound to component timesteps
h_{min}	Lower bound to component timesteps
$\mathbf{Z}(t)$	Vector of numerical state vectors (See Eq. 2.39)
t_k	Current time and time point on a global time grid
$\mathbf{B}^{[*]}$	Constant matrix
$\boldsymbol{\nu}$	Eigenvector
λ	Eigenvalue
r_s	*Splitting ratio* (See Eq. 3.20)
Δt_{s}	Smaller timestep for updating fast moving forcing component vector
Δt_ℓ	Larger timestep for updating slow moving forcing component vector
n	*Timestep ratio i.e* Ratio of larger timestep to smaller timestep
μ_{s}	Scalar parameter connected to the smaller timestep
μ_ℓ	Scalar parameter connected to the larger timestep
ξ	Damping ratio
ϵ	Global error
$\mathcal{K}^{[*]}$	Set of indices (See Eq. 4.2)
$\mathfrak{S}^{[*]}$	Set of indices (See Eq. 4.1)
$\mathcal{A}^{[*]}$	Constant matrix (See Eq. 4.5)
$\phi^{[*]}$	Local component state vector
$\mathbf{f}^{[*]}$	Local component force vector

j_k	Index of the component force vector updated at time t_k (See Eq. 2.5)
$\mathbf{L}_m^{\Delta t}$	Local discretization error in m^{th} order approximation
Γ	Mapping function of exact solution
$\mathbf{F}^{(m)}$	m^{th} order time differential of force vector
δ_{total}	Cost of simulation due to force vector evaluations (See Eq. 2.79)
δ	Cost of single force vector evaluation (See Eq. 2.79)

Chapter 1

Introduction

Many engineered and natural systems involve multiple characteristic timescales, which present difficulties for efficient numerical simulation. These multiple-time-scale systems can be broadly classified into two classes: (1) systems with *discrete time scales*, such as molecular dynamics [26] and electrical networks [38], and (2) systems with *continuously-distributed time scales*, such as aerosol dynamics [74], multiscale structural dynamics [3], and turbulent fluid flow [9]. In discrete time scale systems where there is a clear separation of scales into fast and slow, there are many techniques available [29, 19, 49], both analytical and numerical, to average or homogenize the fast timescales, leaving a system with only the slow timescales to be simulated. In cases where this is not possible, either because the fast timescales are important, because of system complexity or the fact that there is simply a wide range of timescales with no clear separation, such as the continuously-distributed time scales systems, it has traditionally been necessary to simulate the entire system at the rate of the fastest timescale, which can be very expensive. Examples of systems where this occurs include finite elements with non-uniform element meshes [44], molecular dynamics [73], aerosol condensation/reaction dynamics [74] and turbulent fluid flow [9].

The first attempt to use more than a single timestep to address multiple-time-scale problems was proposed by Tildesley *et. al.* [66] in the field of astronomy. Ever since, much effort has been devoted in developing methods addressing *discrete time scale* problems which include *multiple time-stepping* (MTS) methods [48, 14, 30], which allow different timesteps to be used for integrating different components of the system. They were first proposed for use in astronomy [31], were later used for studying molecular dynamics [65],

and have been more recently used in structural dynamic applications [15, 62, 23]. In the context of molecular dynamics, r-RESPA [26, 71] is perhaps the most widely known MTS method. MTS methods are known to have resonance instabilities, especially in the context of molecular dynamic simulations [54, 63], which can severely limit the computational gain achieved. Several approaches have been proposed to reduce these instabilities [63, 47]. Unlike in the case of molecular dynamics simulations, MTS methods in structural dynamics are less hampered by resonance instabilities. Several studies have shown that the timesteps leading to resonance instabilities are far less likely to be chosen in practice [14, 12, 13]. Hughes *et al.* [33] have proven convergence for MTS methods and have shown that MTS have at most first order convergence. Some efforts have been made to combine MTS with linear multi-step methods, also known as *multirate methods* [51, 10, 21], but were found to be at most second order accurate because of interface treatments or coupling terms [21, 38, 4]. Recently, an asynchronous version of MTS has also been investigated in the context of atmospheric applications [74].

A second class of numerical methods for multiple-time-scale systems are based on space-time finite element formulations. These were first proposed by Hughes *et al.* [34, 35] based on the use of time-discontinuous Galerkin methods, which allowed for unstructured meshes and gave a framework for adaptive schemes, including rigorous error estimates and convergence proofs. Fully discontinuous Galerkin methods in spacetime have also been investigated [3], which allow discontinuities in both time and space and have been applied to shock-wave propagation, boundary tracking, crystalline solid dynamics [39], fracture propagation [2], and other areas. They have been shown to have a high order of convergence, depending upon the polynomial degree of the spacetime finite element basis functions, but require spacetime mesh generation algorithms [16, 1]. Added to that, the computational gain achieved can be limited because the variation in the timesteps chosen may not be very large.

A third class of methods proposed are for addressing the problem of distributed simulation of multibody systems [72]. These works address the problem of providing a simulation

environment, also known as *gluing*-algorithm [67], where heterogenous subsystem mechanical system models are coupled together to perform dynamic simulation of the entire system, while the subsystem mechanical system models are allowed to develop independently. As a result, each subsystem can use its own integration timestep and the global glue algorithm enforces the constraints at the interface at desired time intervals. Several such algorithms are proposed and studied in [72, 69, 68, 67] and are used to model complex mechanical systems such as a complete automotive vehicle.

A fourth class of multiple-time-step numerical methods are Asynchronous Variational Integrators (AVI) [44] and their generalization to Asynchronous Splitting Methods (ASM) [53, 45]. AVIs are a class of integrators derived from the discretization of Hamilton's variational principle in Lagrangian mechanics and permit the use of independent timesteps for different system components, unlike in multiple time-stepping methods where timestep ratios are integer multiples of each other. As a consequence of their variational structure, AVI methods conserve local momenta and a local discrete multisymplectic structure exactly. However, the convergence of AVI methods, and the more general ASM integrators, which has been proven [45], are only second order. In addition, they have also been shown that they can exhibit resonance instabilities similar to those of MTS methods [17], which can limit their computational gain in some circumstances.

To understand stability-related issues with the multiple-time-step methods as well as *asynchronous* methods, we further classify then into *extrapolative* [6, 5] and *impulse* [24, 7, 18, 71] based methods. Impulse-based schemes update the impact of slower force vectors in the form of impulses while integrating the fast force vector more frequently. As a result they can preserve the symplectic structure and hence are *symplectic* methods. However, these methods generally suffer from resonance instabilities whenever the timestep of the slower force vector is commensurate with the time period of the faster time scale in the system [17, 5, 52]. As a result, for some systems the efficiency gain compared to traditional synchronous methods is limited. Extrapolative schemes use force extrapolations of slower

variables while updating the faster variables and therefore can be shown to not conserve symplectic structures. While these methods are not known to suffer from resonance related stability limitations of impulse based methods, they suffer from energy drift issues [5, 6].

In this work, a new numerical solution method for multiple-time-scale systems is introduced, which is arbitrarily high-order and does not suffer from resonance instabilities. Extending the work of Shin [61], we developed a one parameter family of high-order Multistep based Asynchronous Splitting Methods (MASM), which generalizes Asynchronous Splitting Methods (ASM) to higher-order methods, based on an asynchronous version of classical multistep methods [28]. MASM are explicit, have order m for m-steps, and are cheap to evaluate at each step. The MASM methods are shown to be convergent, stable and have higher efficiency than the corresponding synchronous methods.

For practical implementation of numerical ODE solvers, it is typically important to have adaptive time-step and order selection. These ideas were first popularized by Krogh's DVDQ [41] and Gear's DIFFSUB [22] codes. Present-day ODE solvers rely heavily on these works, including examples such as *ode113* [59] in MATLAB. We develop a time-step selection (TSS) method for solving ODEs using MASM. There are many such variable timestep implementations [43, 40, 42, 56, 64, 59, 8, 57], but we here only explored a basic implementation of TSS using MASM. We also present a sparse implementation of MASM useful for large scale simulations as well as parallel implementation.

The rest of the thesis is organized as follows. In Chapter 2, we review ASM and linear multistep methods, and define the new MASM integrators, including the derivation of the coefficient conditions for maximal order, as well as the proof of convergence and some numerical studies. In Chapter 3, we look at the stability of MASM, which is studied analytically and numerically. Chapter 4 discusses the timestep selection strategy developed for using with MASM, including the results of numerical experiments studying the performance of the same. Finally, in Chapter 5 we present our conclusions.

Chapter 2

MASM and its Convergence

2.1 MASM

In this chapter, before we formulate MASM, we briefly review linear multistep methods and the ASM methods that have inspired MASM. In this thesis, we restrict ourselves to Adams-type MASM, which corresponds to Adams-type multistep methods, i.e., *Adams-Bashforth Methods* (ABM). We also establish the conditions for achieving maximal order. We prove a few important results before proving the convergence of MASM and explore this convergence via numerical experiments.

2.1.1 Linear multistep methods and Adams-Bashforth Methods (ABM)

We begin by recalling the classical theory of linear multistep methods. We will consider systems of first-order ordinary differential equations (ODEs) written as

$$\dot{\mathbf{X}}(t) = \mathbf{F}(\mathbf{X}, t), \tag{2.1}$$

where $\mathbf{X}(t) \in \mathbb{R}^M$ is the state of the system at time t and we assume that $\mathbf{F}$ is Lipschitz.

An explicit m-step general linear multistep method [28] for solving (2.1) is given by

$$\mathbf{\Phi}_{k+1} + \sum_{i=0}^{m-1} \alpha_i \mathbf{\Phi}_{k-i} = \Delta t_k \sum_{j=0}^{m-1} \beta_j \mathbf{F}_{k-j}, \tag{2.2}$$

where $\mathbf{\Phi}_k$ approximates $\mathbf{X}(t_k)$, $\mathbf{F}_k = \mathbf{F}(\mathbf{\Phi}_k, t_k)$ and $\Delta t_k = t_{k+1} - t_k$.

From the above equation it can be seen that the multistep scheme uses the history of the state as well as the history of function evaluations to determine the state at the next timestep. The coefficients α_i and β_i determine the consistency, order, and stability of the method (see, e.g., Hairer *et al.* [28] for details). The primary advantage of multistep methods is that they are very cheap to evaluate, requiring only a single function evaluation per step, in contrast to m-stage Runge Kutta methods, for example, which require m function evaluations per step.

We will be particularly interested in the *Adams-Bashforth Methods (ABM)*, a family of linear multistep methods. These methods have $\alpha_0 = -1$ and $\alpha_1 = \alpha_2 = \cdots = \alpha_{m-1} = 0$, giving the m-step ABM to be

$$\mathbf{\Phi}_{k+1} = \mathbf{\Phi}_k + \Delta t_k \sum_{j=0}^{m-1} \beta_j \mathbf{F}_{k-j}, \tag{2.3}$$

and the coefficients β_i are uniquely determined by the requirement that the method be of order m (see, e.g., Iserles [37]). Such methods are stable [28, Chapter III.3] and of maximal order m, as shown by the first Dahlquist barrier result [28, Theorem III.3.5]. Note that the second Dahlquist barrier shows that explicit linear multistep methods cannot be A-stable.

Multistep methods require a *starting procedure* to generate the first $m - 1$ steps, before the equation (2.2) can be used. This can be done using a high-order one-step method, such as a Runge-Kutta method, or, more commonly, by using low-order multistep methods with very small stepsizes [28, Chapter III.1]. It is also common to use adaptive stepsize and order selection [28, Chapter III.7] to compute the solution with the least cost for a given error tolerance. To obtain the solution at particular times, perhaps including the final time, an appropriate-order interpolation can be used.

2.1.2 Asynchronous Splitting Methods (ASM)

We say that a *splitting* [46] of ODE (2.1) is a decomposition of the form

$$\dot{\mathbf{X}}(t) = \mathbf{F}(\mathbf{X}, t) = \sum_{j=1}^{N_{\mathrm{f}}} \mathbf{F}^{[j]}(\mathbf{X}, t) \tag{2.4}$$

for *component* Lipschitz functions $\mathbf{F}^{[j]}$. Note that this corresponds to a splitting of the vector field, not a splitting of the variables. Multiscale numerical methods based on fast/slow variable splittings have been widely explored, but are different in spirit to vector field splittings. A *splitting method* for (2.4) composes integrators for each component $\mathbf{F}^{[j]}$ to give an integrator for the entire system (see McLachlan and Quispel [46] for a overview). This allows the integration method to be designed for each component.

Asynchronous Splitting Methods (ASM) [45] generalize splitting methods by allowing different timesteps to be used for each component $\mathbf{F}^{[j]}$. Given one-step integrators $\Psi_{\Delta t}^{[j]} : \mathbb{R}^M \to \mathbb{R}^M$ for the component vector field $\mathbf{F}^{[j]}$, together with a sequence of timesteps Δt_k and indices j_k, the method is defined by

$$\mathbf{\Phi}_{k+1} = \Psi_{\Delta t_k}^{[j_k]}(\mathbf{\Phi}_k). \tag{2.5}$$

That is, the kth integration step consists of applying the integrator for component j_k for timestep Δt_k. We define the *cumulative time* for component j at step k to be

$$t_k^{[j]} = \sum_{p=1}^{k} \delta_{j,j_p} \Delta t_p \tag{2.6}$$

where $\delta_{i,j}$ is *Kronecker delta* function and the *global minimum time* to be

$$t_k^{\min} = \min_{j=1,\ldots,N_{\mathrm{f}}} t_k^{[j]}. \tag{2.7}$$

See Figure 2.1 (left panel) for a graphical illustration of these definitions and Lew *et. al.* [45]

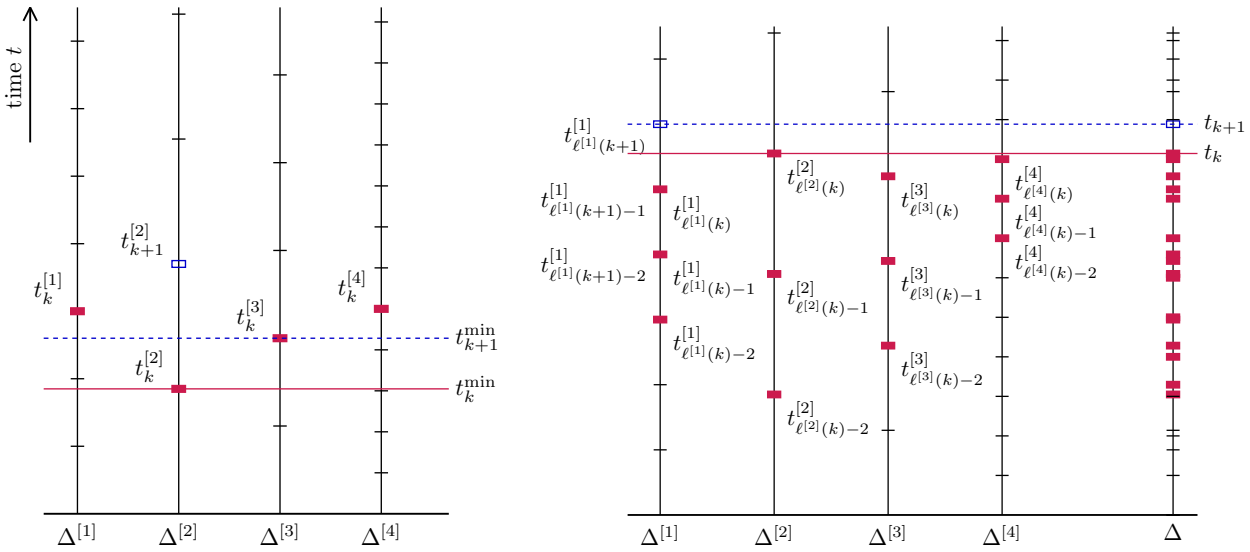

Figure 2.1: Left: Asynchronous Splitting Method (ASM) time variables. Right: Multistep ASM (MASM) time variables for 3-step MASM. The two methods are not depicted at the same times.

for a discussion of algorithm implementation.

The *maximum asynchronicity* is the greatest time by which any vector field component is advanced in time more than other component, given by

$$\Delta t_{\text{asynch}} = \max_k \left\{ \max_{j \in \{1,\dots,N_{\text{f}}\}} \left(t_k^{[j]} - t_k^{\min} \right) \right\}. \tag{2.8}$$

It is proved in Lew *et al.* [45, Theorem A.7] that an ASM is convergent as $\Delta t_{\text{asynch}} \to 0$, assuming that each component integrator $\Psi^{[j]}$ is consistent. In general this convergence is of first order in the timestep scaling.

2.1.3 Multistep Asynchronous Splitting Methods (MASM)

We will work with a split vector field of the form (2.4) and generalize linear multistep methods (2.2) to Multistep Asynchronous Splitting Methods (MASM). To do this, we first introduce some additional notation.

In time integration, different time grids

$$\Delta^{[j]} := \{t_0^{[j]} = t_0 < t_1^{[j]} < \ldots < t_p^{[j]} < t_{p+1}^{[j]} < \ldots < t_{N^{[j]}}^{[j]} = t_{end}\}, \quad (j = 1, \ldots, N_{\rm f}) \tag{2.9}$$

are considered for each individual term $\mathbf{F}^{[j]}$. The full splitting method uses the grid

$$\Delta := \{t_0 < t_1 < \ldots < t_k < t_{k+1} < \ldots < t_N = t_{end}\} := \bigcup_{j=1}^{N_{\rm f}} \Delta^{[j]}. \tag{2.10}$$

An m-step *explicit linear Multistep Asynchronous Splitting Method (MASM)* integrator proceeds in timesteps $t_k \leftarrow t_k + \Delta t_k$ updating the numerical solution $\mathbf{\Phi}_k \approx \mathbf{X}(t_k)$ by

$$\mathbf{\Phi}_{k+1} = \mathbf{\Phi}_k + \Delta t_k \sum_{j=1}^{N_{\rm f}} \sum_{i=0}^{M_k-1} \left\langle \tilde{\boldsymbol{\beta}}_k^{[j]} \right\rangle_i \mathbf{F}^{[j]}(\mathbf{\Phi}_{k-i}, t_{k-i}) \tag{2.11}$$

with $\left\langle \tilde{\boldsymbol{\beta}}_k^{[j]} \right\rangle_i$=0 if $t_{k-i} \notin \Delta^{[j]}$. Also note that because of the condition on $\left\langle \tilde{\boldsymbol{\beta}}_k^{[j]} \right\rangle_i$, $\mathbf{F}^{[j]}$ need not be evaluated at all times $t_k \in \Delta$ but instead are only evaluated at times $t_k \in \Delta^{[j]}$.

In an m-step method, there are for any k ≥ 0 and $j \in \{1, \ldots, N_{\rm f}\}$ at most m non-vanishing coefficients $\left\langle \tilde{\boldsymbol{\beta}}_k^{[j]} \right\rangle_i$, $(i = 0, 1, \ldots, M_k - 1)$. More precisely: Let

$$\{t_p^{[j]} \in \Delta^{[j]} : t_p^{[j]} \leq t_k\} = \{t_0^{[j]} < t_1^{[j]} < \ldots < t_{\ell^{[j]}(k)}^{[j]}\}, \quad \textit{i.e.,} \quad t_{\ell^{[j]}(k)}^{[j]} = \max\{t_p^{[j]} \in \Delta^{[j]} : t_p^{[j]} \leq t_k\} \tag{2.12}$$

then

$$\left\langle \tilde{\boldsymbol{\beta}}_k^{[j]} \right\rangle_i = 0 \text{ if } t_{k-i} \notin \{t_{\ell^{[j]}(k)}^{[j]}, t_{\ell^{[j]}(k)-1}^{[j]}, \ldots, t_{\ell^{[j]}(k)-(m-1)}^{[j]}\}. \tag{2.13}$$

The number M_k of terms in the inner sum of (2.11) is given implicitly by

$$t_{k-(M_k-1)} = \min_{j=1,\ldots,N_{\rm f}} t_{\ell^{[j]}(k)-(m-1)}^{[j]}. \tag{2.14}$$

The *current system time* is

$$t_k = \max_{j=1,\ldots,N_\mathrm{f}} t^{[j]}_{\ell^{[j]}(k)} \tag{2.15}$$

and the *current system state* is

$$\boldsymbol{\Phi}_k (\approx \mathbf{X}(t^{[j]}_{\ell^{[j]}(k)})) \text{ for } j \text{ chosen such that } t_k = t^{[j]}_{\ell^{[j]}(k)}. \tag{2.16}$$

These variables are illustrated in Figure 2.1 (right panel).

Given *component timesteps* $\Delta t^{[j]}_{\ell^{[j]}(k)}$ for each component j, at each time t_k we choose the *global timestep* to be

$$\Delta t_k = \min_{j=1,\ldots,N_\mathrm{f}} \left(t^{[j]}_{\ell^{[j]}(k)} + \Delta t^{[j]}_{\ell^{[j]}(k)} - t_k \right). \tag{2.17}$$

Note that this is different from the timestep definition (2.6) for ASM. We may update each component timestep within the time interval described by that timestep itself, subject to the condition that $\Delta t^{[j]}_{\ell^{[j]}(k+1)} > t_{k+1} - t^{[j]}_{\ell^{[j]}(k)}$ if $t_{k+1} < t^{[j]}_{\ell^{[j]}(k)} + \Delta t^{[j]}_{\ell^{[j]}(k)}$, to ensure that time only advances forward.

As with a regular Adams-Bashforth multistep method (2.3), we need a *starting procedure* to compute the first $m-1$ steps for each component j. We use a high-order one-step method to do this, but very small stepsizes with a lower-order multistep method would also be possible [28, Chapter III.7]. A *self* start algorithm using adaptive timestep and order selection can also be employed [50, 58], which is also discussed later in Chapter 4. We do use a standard interpolation to evaluate the solution at particular times, including the final time.

The coefficients $\left\langle \tilde{\boldsymbol{\beta}}^{[j]}_k \right\rangle_i$ must generally be determined for each component at each step, also described in the next section, as the local incremental step sizes are varying. The data that must be maintained for MASM (2.11) is the current state $\boldsymbol{\Phi}_k$ and the m previous function evaluations $\mathbf{F}^{[j]}(\boldsymbol{\Phi}^{[j]}_{\ell^{[j]}(k)-i}, t^{[j]}_{\ell^{[j]}(k)-i})$, $i = 0, \ldots, m-1$, for each system component

$j \in \{1, \ldots, N_\mathrm{f}\}$. For fully-coupled ODEs this could be substantially more expensive in storage than a traditional *synchronous* multistep method, but for sparsely coupled systems (such as that in Section 2.4.2) this may be only slightly more expensive in terms of storage.

The algorithmic description of MASM is given in Algorithm 1 and a graphical depiction of the variable indexing is shown in Figure 2.1 (right panel). Note that Algorithm 1 is written without in-place updating of variables (except k). In practical implementations only single $m \times N_\mathrm{f}$ arrays are stored for times $t^{[j]}_{\ell^{[j]}(k)-i}$, and force vector field $\mathbf{F}^{[j]}_{\ell^{[j]}(k)-i}$, where $i = 0, \ldots, m-1$, and $j = 1, \ldots, N_\mathrm{f}$ and these are updated in-place. In case $\{\alpha_i\}_{i=1}^{m-1} \neq 0$ (see Eq. 2.2), one would also need to store history of the states, $\mathbf{\Phi}^{[j]}_{\ell^{[j]}(k)-i}$ ($\approx \mathbf{X}(t^{[j]}_{\ell^{[j]}(k)-i})$). In addition, if the system components are sparsely coupled (such as in Section 2.4.2), it may be unnecessary to update all components at each step.

2.2 Order Condition

We follow Hairer *et al.* [28, Chapter III.2] for definitions of local error and method order.

Definition 1. *The* local error *of a MASM (2.11) is defined by*

$$\mathbf{X}(t_{k+1}) - \mathbf{\Phi}_{k+1} \tag{2.18}$$

where $\mathbf{X}(t)$ is the exact solution of the ODE (2.4) and $\mathbf{\Phi}_{k+1}$ is the numerical solution obtained from (2.11) with the previous states set to the exact values $\mathbf{\Phi}_p = \mathbf{X}(t_p)$ for $p = 1, 2, \ldots, k$.

While it would also be possible to introduce an appropriate linear difference operator and generating polynomials in the MASM framework, here we consider only the most simple definition of method order, as follows.

Definition 2. *The MASM (2.11) is said of be of* order *n if the local error (2.18) is $\mathcal{O}(\Delta t_k)^{n+1}$ for all sufficiently regular ODEs (2.4), where all other timesteps are defined to be constant-factor multiples of Δt_k.*

We will focus in this thesis on Adams-type MASM integrators (2.11). An m-step MASM of this type can be of order m, and this condition uniquely determines the coefficients $\left\langle \tilde{\boldsymbol{\beta}}_k^{[j]} \right\rangle_i$ as shown in the following theorem.

Theorem 1. *The m-step Adams-type MASM (2.11) is of order m if the coefficients $\left\langle \boldsymbol{\beta}_k^{[j]} \right\rangle_i$ satisfy*

$$\mathbf{V}_k^{[j]} \boldsymbol{\beta}_k^{[j]} = \mathbf{R} \tag{2.19}$$

for the Vandermonde matrix $\mathbf{V}_k^{[j]}$, vector $\mathbf{R}$ and vector $\boldsymbol{\beta}_k^{[j]}$ defined by

$$\left[\mathbf{V}_k^{[j]}\right]_{p+1,i+1} = \left(\tau^{[j]}_{\ell^{[j]}(k)-i}\right)^p \quad \text{for } p, i = 0, \ldots, m-1 \tag{2.20}$$

$$\langle \mathbf{R} \rangle_{p+1} = \frac{1}{p+1} \text{ for } p = 0, \ldots, m-1, \tag{2.21}$$

$$\left\langle \boldsymbol{\beta}_k^{[j]} \right\rangle_{q+1} \triangleq \left\langle \tilde{\boldsymbol{\beta}}_k^{[j]} \right\rangle_{s+1} \text{ where } t^{[j]}_{\ell^{[j]}(k)-q} = t_{k-s} \, (\textit{Note: } s \in \{0, \ldots, M_k - 1\} \textit{ and } q \in \{0, \ldots, m-1\}), \tag{2.22}$$

where $\tau^{[j]}_{\ell^{[j]}(k)-i} = (t^{[j]}_{\ell^{[j]}(k)-i} - t_k)/\Delta t_k$. Furthermore, the coefficients $\left\langle \boldsymbol{\beta}_k^{[j]} \right\rangle_i$ exist and are uniquely determined so long as all timesteps are positive.

Proof. Taking $\mathbf{X}(t)$ and $\boldsymbol{\Phi}_p$ as in Definition 1, using Eq. (2.22) one can reduce Eq. (2.11) to;

$$\boldsymbol{\Phi}_{k+1} = \mathbf{X}(t_k) + \Delta t_k \sum_{i=0}^{m-1} \sum_{j=1}^{N_\mathrm{f}} \left\langle \boldsymbol{\beta}_k^{[j]} \right\rangle_i \mathbf{F}^{[j]}(\mathbf{X}(t^{[j]}_{\ell^{[j]}(k)-i}), t^{[j]}_{\ell^{[j]}(k)-i}) \tag{2.23}$$

$$= \mathbf{X}(t_k) + \Delta t_k \sum_{i=0}^{m-1} \sum_{j=1}^{N_\mathrm{f}} \left\langle \boldsymbol{\beta}_k^{[j]} \right\rangle_i \sum_{p=0}^{m-1} \frac{d^p \mathbf{F}^{[j]}(\mathbf{X}(t_k), t_k)}{dt^p} \frac{(t^{[j]}_{\ell^{[j]}(k)-i} - t_k)^p}{p!} + \mathcal{O}(\Delta t_k^{m+1}) \tag{2.24}$$

$$= \mathbf{X}(t_k) + \sum_{p=0}^{m-1} \sum_{j=1}^{N_\mathrm{f}} \sum_{i=0}^{m-1} \frac{d^p \mathbf{F}^{[j]}(\mathbf{X}(t_k), t_k)}{dt^p} \frac{(\Delta t_k)^{p+1}}{p!} \left\langle \boldsymbol{\beta}_k^{[j]} \right\rangle_i (\tau^{[j]}_{\ell^{[j]}(k)-i})^p + \mathcal{O}(\Delta t_k^{m+1}). \tag{2.25}$$

Taking a Taylor expansion of $\mathbf{X}(t)$ as the exact solution of (2.4) now gives

$$\mathbf{X}(t_{k+1}) = \mathbf{X}(t_k) + \sum_{p=0}^{m-1} \sum_{j=1}^{N_\mathrm{f}} \frac{d^p \mathbf{F}^{[j]}(\mathbf{X}(t_k), t_k)}{dt^p} \frac{(\Delta t_k)^{p+1}}{(p+1)!} + \mathcal{O}(\Delta t_k^{m+1}). \tag{2.26}$$

Comparing (2.25) and (2.26) we see that the $\left\langle \boldsymbol{\beta}_k^{[j]} \right\rangle_i$ must satisfy

$$\sum_{i=0}^{m-1} \left\langle \boldsymbol{\beta}_k^{[j]} \right\rangle_i (\tau^{[j]}_{\ell^{[j]}(k)-i})^p = \frac{1}{p+1} \tag{2.27}$$

for $p = 0, \ldots, m-1$, which is (2.19). As $\mathbf{V}_k^{[j]}$ is Vandermonde, its determinant is given by

$$\det \mathbf{V}_k^{[j]} = \prod_{0 \le p < i \le m-1} (\tau^{[j]}_{\ell^{[j]}(k)-i} - \tau^{[j]}_{\ell^{[j]}(k)-p}), \tag{2.28}$$

which is nonzero for positive timesteps, and thus the $\left\langle \boldsymbol{\beta}_k^{[j]} \right\rangle_i$ are uniquely determined by (2.19). □ □

The linear system in Theorem 1 that must be solved for the $\left\langle \boldsymbol{\beta}_k^{[j]} \right\rangle_i$ coefficients can be

written out as

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ \tau^{[j]}_{\ell^{[j]}(k)} & \tau^{[j]}_{\ell^{[j]}(k)-1} & \tau^{[j]}_{\ell^{[j]}(k)-2} & \cdots & \tau^{[j]}_{\ell^{[j]}(k)-(m-1)} \\ (\tau^{[j]}_{\ell^{[j]}(k)})^2 & (\tau^{[j]}_{\ell^{[j]}(k)-1})^2 & (\tau^{[j]}_{\ell^{[j]}(k)-2})^2 & \cdots & (\tau^{[j]}_{\ell^{[j]}(k)-(m-1)})^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (\tau^{[j]}_{\ell^{[j]}(k)})^{m-1} & (\tau^{[j]}_{\ell^{[j]}(k)-1})^{m-1} & (\tau^{[j]}_{\ell^{[j]}(k)-2})^{m-1} & \cdots & (\tau^{[j]}_{\ell^{[j]}(k)-(m-1)})^{m-1} \end{bmatrix} \boldsymbol{\beta}^{[j]}_k = \begin{bmatrix} 1 \\ 1/2 \\ 1/3 \\ \vdots \\ 1/m \end{bmatrix}. \tag{2.29}$$

This system can also be obtained by considering the polynomial interpolation problem required to construct an mth order MASM integrator.

For high order MASM integrators the expense of solving the system (2.29) for each component at each timestep can be very significant. For lower order methods ($m \leq 4$, for example) with expensive function evaluations, solving this system is comparatively cheap. Efficient solvers for this system are possible for any m, using the Vandermonde structure [50, Section 2.8.1].

2.3 Convergence

Convergence of MASM integrators can be proved in a similar fashion as to the convergence of linear multistep methods (see [28]), which require expressing MASM (2.11) as a one-step method as an initial step. Unlike the multistep methods, the one-step method definition for MASM is not straight forward as the length of the history of the forcing used, *i.e.* M_k, at each step varies and also could grow with refinement of timesteps. However, if the component timesteps satisfy,

$$\left(\frac{h_{max}}{h_{min}}\right) \leq C_t, \tag{2.30}$$

where C_t is a constant and $\{h_{max}, h_{min}\} = \{\max_{j,k}(\Delta t^{[j]}_{\ell^{[j]}(k)}), \min_{j,k}(\Delta t^{[j]}_{\ell^{[j]}(k)})\}$ for all

$j \in \{1, 2, \ldots, N_\mathrm{f}\}$ and $k \geq 0$, then an upper bound on M_k exists as shown in the following *Lemma*.

Lemma 1. *Given the component timesteps,* $\{\Delta t_p^{[j]}\}$*, of MASM (2.11) satisfy Eq. (2.30), then*

$$M_k \leq (m+1) N_\mathrm{f} C_t, \tag{2.31}$$

for all $k \geq 0$.

Proof. From Eq. (2.14) M_k can be written as,

$$M_k \leq \sum_{j=1}^{N_\mathrm{f}} (s_k^{[j]} + 1) \tag{2.32}$$

where $s_k^{[j]}$ are defined by,

$$s_k^{[j]} = \max \{s | s \in \mathbb{N}^+ \text{ such that } t^{[j]}_{\ell^{[j]}(k)-s} \geq \min_{\hat{j}} t_{\hat{j},\ell^{[\hat{j}]}(k)-(m-1)}\}. \tag{2.33}$$

Note that $s_k^{[j]} + 1 \geq m$ for all $j \in \{1, 2, \ldots, N_\mathrm{f}\}$ and $k \geq 0$. From the definition of $s_k^{[j]}$ (Eqn. 2.33) and using the fact that $t^{[j]}_{\ell^{[j]}(k)-p} = t^{[j]}_{\ell^{[j]}(k)} - \sum_{i=1}^{p} \Delta t^{[j]}_{\ell^{[j]}(k)-i}$, one can easily show that,

$$\sum_{i=0}^{s_k^{[j]}} \Delta t^{[j]}_{\ell^{[j]}(k)-i} \leq t^{[j]}_{\ell^{[j]}(k)+1} - t_{j_m,\ell^{[j_m]}(k)} + \sum_{i=1}^{(m-1)} \Delta t_{j_m,\ell^{[j_m]}(k)-i}, \tag{2.34}$$

where j_m is such that $t_{j_m,\ell^{[j_m]}(k)-(m-1)} = \min_{\hat{j}} t_{\hat{j},\ell^{[\hat{j}]}(k)-(m-1)}$. Since $h_{min} \leq \Delta t_{j_m,\ell^{[j_m]}(k)-i} \leq h_{max}$ and $\max_j t^{[j]}_{\ell^{[j]}(k)+1} - \min_j t^{[j]}_{\ell^{[j]}(k)} \leq 2h_{max}$,

$$(s_k^{[j]} + 1) h_{min} \leq \sum_{i=0}^{s_k^{[j]}} \Delta t^{[j]}_{\ell^{[j]}(k)-i} \leq (m+1) h_{max}, \tag{2.35}$$

$$(s_k^{[j]} + 1) \leq (m+1) \left(\frac{h_{max}}{h_{min}} \right). \tag{2.36}$$

Therefore, recalling Eq. (2.32),

$$M_k \leq \sum_{j=1}^{N_\mathrm{f}} (s_k^{[j]} + 1) \leq (m+1) N_\mathrm{f} \left(\frac{h_{max}}{h_{min}} \right), \tag{2.37}$$

$$M_k \leq (m+1) N_\mathrm{f} C_t. \tag{2.38}$$

Hence proved. □ □

C_t is a measure of the range of times scales in the numerical simulation. Also note that it remains constant when the timesteps of the components are refined uniformly. Once an upper bound for M_k is established and one-step MASM is defined, the proof for convergence of MASM will be very similar to the convergence of linear multistep methods [28], except that it also requires $\left\langle \boldsymbol{\beta}_k^{[j]} \right\rangle_i$ to be bounded above. Below we firstly define the one-step MASM and then prove that $\left\langle \boldsymbol{\beta}_k^{[j]} \right\rangle_i$ are bounded from above.

Let $\mathbf{Z}_k = [\boldsymbol{\Phi}_k^T \ \boldsymbol{\Phi}_{k-1}^T \ \ldots \ \boldsymbol{\Phi}_{k-(M-1)}^T]^T$, using which one can express Eq. (2.11) as,

$$\mathbf{Z}_{k+1} = (\mathbf{A} \otimes \mathbf{I}) \mathbf{Z}_k + \Delta t_k \Lambda_k (\mathbf{Z}_k, \{\Delta t_i\}), \tag{2.39}$$

where the increment function $\Lambda_k(\mathbf{Z}_k, \{\Delta t_i\}_{i=k-(M-1)}^k) = [(\sum_{j=1}^{N_\mathrm{f}} \sum_{i=0}^{M-1} \left\langle \hat{\boldsymbol{\beta}}_k^{[j]} \right\rangle_i \mathbf{F}_{k-i}^{[j]})^T \ \mathbf{0^T} \ \ldots \ \mathbf{0^T}]^\mathbf{T}$, the coefficient matrix is Kronecker tensor product of $\mathbf{I}$, an identity matrix, and $\mathbf{A}$, which is defined by

$$[\mathbf{A}]_{ij} = \begin{cases} 1 & i = j = 1 \\ 1 & i = j+1 \text{ and } j = 1, 2, \ldots, M-2 \\ 0 & \text{otherwise,} \end{cases} \tag{2.40}$$

where $M = (\mathrm{m}+1) N_\mathrm{f} C_t$ (see Eq. 2.38) and

$$\left\langle \hat{\boldsymbol{\beta}}_k^{[j]} \right\rangle_i = \begin{cases} \left\langle \boldsymbol{\beta}_k^{[j]} \right\rangle_p & \text{if } t_{k-i} = t^{[j]}_{\ell^{[j]}(k)-p} \text{ where } p \in \{0, 1, \ldots, m-1\} \\ 0 & \text{elsewhere.} \end{cases} \tag{2.41}$$

Similar to $\left\langle \tilde{\boldsymbol{\beta}}_k^{[j]} \right\rangle_i$ (Eq. 2.13), for each j and k there still remains only m non-zero quantities in $\left\langle \hat{\boldsymbol{\beta}}_k^{[j]} \right\rangle$ and the non-zero quantities are defined by Eq. (2.19). In the following *Lemma* we establish that $\left\langle \hat{\boldsymbol{\beta}}_k^{[j]} \right\rangle_i$ coefficients are bounded above, which will be the last additional result required to prove the convergence of MASM solution.

Lemma 2. *Given* $\{\Delta t_p^{[j]}\}$ *of MASM (2.39) satisfy Eq. (2.30), then there exist a constant* C_β *such that,* $\hat{\boldsymbol{\beta}}_k^{[j]}$ *(2.41) satisfy,* $\left\|\hat{\boldsymbol{\beta}}_k^{[j]}\right\|_\infty \leq C_\beta$, *for all* $j = 1, \ldots, N_\mathrm{f}$ *and* $k \geq 0$.

Proof. Using Eqs. (2.41, 2.19) and also since $\mathbf{R}$ (see Eq. 2.21) is a constant vector with $\|\mathbf{R}\|_\infty = 1$,

$$\left\|\hat{\boldsymbol{\beta}}_k^{[j]}\right\|_\infty = \left\|\boldsymbol{\beta}_k^{[j]}\right\|_\infty = \left\|\left(\mathbf{V}_k^{[j]}\right)^{-1}\mathbf{R}\right\|_\infty \leq \left\|\left(\mathbf{V}_k^{[j]}\right)^{-1}\right\|_\infty \|\mathbf{R}\|_\infty, \tag{2.42}$$

$$\leq \left\|\left(\mathbf{V}_k^{[j]}\right)^{-1}\right\|_\infty. \tag{2.43}$$

Norm of an Inverse *Vandermonde matrix* is known to be bounded above (see [32] pp. 417) and is given by,

$$\left\|\left(\mathbf{V}_k^{[j]}\right)^{-1}\right\|_\infty \leq \max_p \Pi_{q\neq p} \frac{1 + |\tau^{[j]}_{\ell^{[j]}(k)-q}|}{|\tau^{[j]}_{\ell^{[j]}(k)-p} - \tau^{[j]}_{\ell^{[j]}(k)-q}|}. \tag{2.44}$$

To get an estimate for the above upper bound, recall Eq. (2.20),

$$\tau^{[j]}_{\ell^{[j]}(k)-i} = \frac{t^{[j]}_{\ell^{[j]}(k)-i} - t_k}{\Delta t_k} = (-1)\left(\frac{\sum_{r=1}^{i} \Delta t^{[j]}_{\ell^{[j]}(k)-r} + \left(t_k - t^{[j]}_{\ell^{[j]}(k)}\right)}{\Delta t_k}\right). \tag{2.45}$$

Since $\Delta t^{[j]}_{\ell^{[j]}(k)} \geq t_k - t^{[j]}_{\ell^{[j]}(k)} > 0$ (see Eq. 2.15),

$$\left|\left(\frac{\sum_{r=1}^{i}\Delta t^{[j]}_{\ell^{[j]}(k)-r}}{\Delta t_k}\right)\right| < |\tau^{[j]}_{\ell^{[j]}(k)-i}| \leq \left|\left(\frac{\sum_{r=0}^{i}\Delta t^{[j]}_{\ell^{[j]}(k)-r}}{\Delta t_k}\right)\right|, \tag{2.46}$$

$$\left|\left(\frac{ih_{min}}{\Delta t_k}\right)\right| < |\tau^{[j]}_{\ell^{[j]}(k)-i}| \leq \left|\left(\frac{(i+1)h_{max}}{\Delta t_k}\right)\right|, \tag{2.47}$$

where $i = 0, 1, \ldots, (m-1)$. Using similar algebra, one can show that,

$$\left|\tau^{[j]}_{\ell^{[j]}(k)-p} - \tau^{[j]}_{\ell^{[j]}(k)-q}\right| = \left|\left(\frac{\sum_{r=\min(p,q)}^{\max(p,q)}\Delta t^{[j]}_{\ell^{[j]}(k)-r}}{\Delta t_k}\right)\right|. \tag{2.48}$$

Therefore, for all j,

$$|p-q|\frac{h_{min}}{\Delta t_k} \leq |\tau^{[j]}_{\ell^{[j]}(k)-p} - \tau^{[j]}_{\ell^{[j]}(k)-q}| \leq |p-q|\frac{h_{max}}{\Delta t_k}. \tag{2.49}$$

The bounds on Δt_k can be obtained using Eq. (2.17) and also Eq. (2.15), *i.e.*,

$$\Delta t_k = \min_{j=1,\ldots,N_{\rm f}}\left(t^{[j]}_{\ell^{[j]}(k)} + \Delta t^{[j]}_{\ell^{[j]}(k)} - t_k\right) \leq \Delta t^{[j]}_{\ell^{[j]}(k)} + \left(t^{[j]}_{\ell^{[j]}(k)} - t_k\right), \tag{2.50}$$

$$\leq \Delta t^{[j]}_{\ell^{[j]}(k)}, \qquad \text{for } j \in \{1, \ldots, N_{\rm f}\}. \tag{2.51}$$

$$0 < \Delta t_k \leq h_{max}. \tag{2.52}$$

Using Eq. (2.49) and Eq. (2.47),

$$\frac{1+q\frac{h_{min}}{\Delta t_k}}{|p-q|\frac{h_{max}}{\Delta t_k}} \leq \frac{1+|\tau^{[j]}_{\ell^{[j]}(k)-q}|}{|\tau^{[j]}_{\ell^{[j]}(k)-p} - \tau^{[j]}_{\ell^{[j]}(k)-q}|} \leq \frac{1+(q+1)\frac{h_{max}}{\Delta t_k}}{|p-q|\frac{h_{min}}{\Delta t_k}}, \tag{2.53}$$

$$\frac{\Delta t_k + qh_{min}}{|p-q|h_{max}} \leq \frac{1+|\tau^{[j]}_{\ell^{[j]}(k)-q}|}{|\tau^{[j]}_{\ell^{[j]}(k)-p} - \tau^{[j]}_{\ell^{[j]}(k)-q}|} \leq \frac{\Delta t_k + (q+1)h_{max}}{|p-q|h_{min}}. \tag{2.54}$$

Using Eqs. (2.52, 2.30), and also since $p \neq q = \{0, 1, \ldots, m-1\}$,

$$\frac{1}{m-2}\frac{1}{C_t} < \frac{1 + |\tau^{[j]}_{\ell^{[j]}(k)-q}|}{|\tau^{[j]}_{\ell^{[j]}(k)-p} - \tau^{[j]}_{\ell^{[j]}(k)-q}|} \leq (m+1)C_t. \tag{2.55}$$

Now from Eqs. (2.43, 2.44, 2.55),

$$\left\|\hat{\boldsymbol{\beta}}^{[j]}_k\right\|_\infty \leq C_\beta, \tag{2.56}$$

where $C_\beta = (m+1)^{m-1}C_t^{m-1}$. Hence proved. □

□

Definition 3. *The* local error *of a one-step MASM (2.39) at t_k is defined by*

$$\mathbf{Z}(t_k) - \mathbf{Z}_k, \tag{2.57}$$

where $\mathbf{Z}(t_k)$ is the exact solution of the ODE (2.4) and $\mathbf{Z}_k$ $(= [\mathbf{\Phi}_k^T\ \mathbf{X}(t_{k-1})^T \ldots \mathbf{X}(t_{k-(M-1)})^T]^T)$ is obtained from (2.39) with $\mathbf{Z}_{k-1} = \mathbf{Z}(t_{k-1}) = [\mathbf{X}(t_{k-1})^T\mathbf{X}(t_{k-2})^T \ldots \mathbf{X}(t_{k-M})^T]^T$.

One can easily show that the vector norm of the local error in one-step MASM is equal to the vector norm of the local error of MASM (Definition 1), i.e. $\|\mathbf{Z}(t_k) - \mathbf{Z}_k\| = \|\mathbf{X}(t_k) - \mathbf{\Phi}_k\|$.

Now following an approach similar to *Lemma 4.3* (section III.4, [28]) and *Theorem 5.8* (section III.5, [28]), convergence of MASM solution is proved in the following theorem and later it is shown to have a order of convergence of atleast m for sufficiently regular ODEs (2.4).

Theorem 2. *Given $\mathbf{F}^{[j]}$ (2.39) are Lipschitz functions with Lipschitz constant L for all $j \in \{1, \ldots, N_\mathrm{f}\}$ and $\{\Delta t_p^{[j]}\}$ satisfy Eq. (2.30), then the global error in MASM (2.39) at time T $(= t_{k+1})$ satisfy,*

$$\|\mathbf{Z}_{k+1}-\mathbf{Z}(t_{k+1})\| \leq \|(\mathbf{A}\otimes\mathbf{I})\|\,\|\mathbf{Z}_k-\mathbf{Z}(t_k)\|+\Delta t_k N_{\mathrm{f}} L M C_\beta \max_{i=0,\ldots,M-1}\|\mathbf{\Phi}_{k-i}-\mathbf{X}(t_{k-i})\|+\|\mathbf{l}_k\| \tag{2.58}$$

where $\mathbf{l}_k = \mathbf{Z}(t_{k+1})-(\mathbf{A}\otimes\mathbf{I})\mathbf{Z}(t_k)-\Delta t_k\Lambda_k(\mathbf{Z}(t_k),\{\Delta t_i\})$, C_β *is a constant (see Eq. 2.56) and* $\|.\|$ *is a vector norm.*

Proof. Recalling Eq. (2.39) and using the definition of $\mathbf{l}_k$, norm of the *global error* at T $(= t_{k+1}-t_0)$ can be written as,

$$\|\mathbf{Z}_{k+1}-\mathbf{Z}(t_{k+1})\| = \|(\mathbf{A}\otimes\mathbf{I})\,(\mathbf{Z}_k-\mathbf{Z}(t_k))+\Delta t_k\,(\Lambda_k(\mathbf{Z}_k,\{\Delta t_i\})-\Lambda_k(\mathbf{Z}(t_k),\{\Delta t_i\}))+\mathbf{l}_k\| \tag{2.59}$$

$$= \|(\mathbf{A}\otimes\mathbf{I})\,(\mathbf{Z}_k-\mathbf{Z}(t_k)) \tag{2.60}$$
$$+\Delta t_k\sum_{j=1}^{N_{\mathrm{f}}}\sum_{i=0}^{M-1}\left\langle\hat{\boldsymbol{\beta}}_k^{[j]}\right\rangle_i\left(\mathbf{F}_{k-i}^{[j]}-\mathbf{F}^{[j]}(\mathbf{X}(t_{k-i}),t_{k-i})\right)+\mathbf{l}_k\|.$$

Since $\mathbf{F}^{[j]}$ is uniformly Lipschitz continuous, where L is the Lipschitz constant, the above equation can be written as,

$$\|\mathbf{Z}_{k+1}-\mathbf{Z}(t_{k+1})\| \leq \|(\mathbf{A}\otimes\mathbf{I})\|\,\|\mathbf{Z}_k-\mathbf{Z}(t_k)\|+\Delta t_k L\sum_{j=1}^{N_{\mathrm{f}}}\sum_{i=0}^{M-1}\left\|\hat{\boldsymbol{\beta}}_k^{[j]}\right\|_\infty\|\mathbf{\Phi}_{k-i}-\mathbf{X}(t_{k-i})\|+\|\mathbf{l}_k\|, \tag{2.61}$$

$$\leq \|(\mathbf{A}\otimes\mathbf{I})\|\,\|\mathbf{Z}_k-\mathbf{Z}(t_k)\| \tag{2.62}$$
$$+\Delta t_k L\sum_{j=1}^{N_{\mathrm{f}}}\left(\sum_{i=0}^{M-1}\left\|\hat{\boldsymbol{\beta}}_k^{[j]}\right\|_\infty\right)\left(\max_{\hat{i}=0,\ldots,M-1}\|\mathbf{\Phi}_{k-\hat{i}}-\mathbf{X}(t_{k-\hat{i}})\|\right)+\|\mathbf{l}_k\|.$$

Invoking Eq. (2.30) and applying *Lemma* 2 one can show that $\left(\sum_{i=0}^{M-1}\left\|\hat{\boldsymbol{\beta}}_k^{[j]}\right\|_\infty\right)\leq MC_\beta$. As a result the above equation can be rewritten as,

$$\|\mathbf{Z}_{k+1} - \mathbf{Z}(t_{k+1})\| \leq \|(\mathbf{A} \otimes \mathbf{I})\| \|\mathbf{Z}_k - \mathbf{Z}(t_k)\| + \Delta t_k N_{\mathrm{f}} L M C_\beta \max_{\hat{i}=0,\ldots,M-1} \|\mathbf{\Phi}_{k-\hat{i}} - \mathbf{X}(t_{k-\hat{i}})\| + \|\mathbf{l}_k\|. \tag{2.63}$$

Hence proved. □

□

The above theorem proves the convergence of the solution of MASM (2.39) as $h_{max} \to 0$.

Theorem 3. *Given* (i) $\{\Delta t_p^{[j]}\}$ *satisfy Eq.* (2.30), (ii) $\mathbf{F}^{[j]}$ (2.39) *are Lipschitz functions with Lipschitz constant* L *for all* $j \in \{1, \ldots, N_{\mathrm{f}}\}$ *and sufficiently differentiable, and* (iii) *the local error (defined by Def.* 3*) in MASM* (2.39) *at k=0 satisfy,* $\|\mathbf{Z}_0 - \mathbf{Z}(t_0)\| \approx O\left(h_{max}\right)^{m+1}$*, then the global error in MASM* (2.39) *at* T $(= t_{k+1})$ *satisfy,*

$$\|\mathbf{Z}_{k+1} - \mathbf{Z}(t_{k+1})\| < C\left(h_{max}\right)^m, \tag{2.64}$$

where $\|.\|$ *is a vector norm.*

Proof. Firstly, we invoke *Theorem* 2,

$$\|\mathbf{Z}_{k+1} - \mathbf{Z}(t_{k+1})\| \leq \|(\mathbf{A} \otimes \mathbf{I})\| \, \|\mathbf{Z}_k - \mathbf{Z}(t_k)\| + \Delta t_k N_{\mathrm{f}} L M C_\beta \max_{i=0,\ldots,M-1} \|\mathbf{\Phi}_{k-i} - \mathbf{X}(t_{k-i})\| + \|\mathbf{l}_k\|. \tag{2.65}$$

Now recall from Eq. (2.40) that $(\mathbf{A} \otimes \mathbf{I})$ is a constant matrix and from *Lemma* 4.4 (section III.4, [28]) there exists a vector norm such that $\|\mathbf{A} \otimes \mathbf{I}\| \leq 1$. Using this result along with the fact that $\max_{i=0,\ldots,M-1} \|\mathbf{\Phi}_{k-i} - \mathbf{X}(t_{k-i})\| \leq \|\mathbf{Z}_k - \mathbf{Z}(t_k)\|$, the *global error* at t_k, can be shown to be,

$$\|\mathbf{Z}_{k+1} - \mathbf{Z}(t_{k+1})\| \leq (1 + \Delta t_k N_{\mathrm{f}} LMC_\beta) \|\mathbf{Z}_k - \mathbf{Z}(t_k)\| + \|\mathbf{l}_k\| \tag{2.66}$$

$$\leq \left(\Pi_{p=0}^{k} (1 + \Delta t_p N_{\mathrm{f}} LMC_\beta)\right) \|\mathbf{Z}_0 - \mathbf{Z}(t_0)\|$$

$$+ (\|\mathbf{l}_k\| + (1 + \Delta t_k N_{\mathrm{f}} LMC_\beta) \|\mathbf{l}_{k-1}\| + \ldots \tag{2.67}$$

$$+ \Pi_{q=1}^{k} (1 + \Delta t_q N_{\mathrm{f}} LMC_\beta) \|\mathbf{l}_0\|) ,$$

since $(1 + \Delta t_p N_{\mathrm{f}} LMC_\beta) < \exp \Delta t_p N_{\mathrm{f}} LMC_\beta$

$$< \exp\left(\left(\sum_{p=0}^{k} \Delta t_p\right) N_{\mathrm{f}} LMC_\beta\right) \|\mathbf{Z}_0 - \mathbf{Z}(t_0)\|$$

$$+ (\|\mathbf{l}_k\| + \exp(\Delta t_k N_{\mathrm{f}} LMC_\beta) \|\mathbf{l}_{k-1}\| + \ldots \tag{2.68}$$

$$+ \exp\left(\left(\sum_{q=1}^{k} \Delta t_q\right) N_{\mathrm{f}} LMC_\beta\right) \|\mathbf{l}_0\|\Bigg) \tag{2.69}$$

$$< \exp((t_{k+1} - t_0) N_{\mathrm{f}} LMC_\beta) \{\|\mathbf{Z}_0 - \mathbf{Z}(t_0)\| + (\|\mathbf{l}_k\| + \|\mathbf{l}_{k-1}\| + \ldots + \|\mathbf{l}_0\|)\} . \tag{2.70}$$

Since $\mathbf{F}^{[j]}$ are sufficiently differentiable, from Definition 2 it is clear that $\|\mathbf{l}_k\| \approx O(\Delta t_k)^{m+1}$ (*i.e.*, there exists constant C_0 such that $\|\mathbf{l}_k\| \leq C_0 (\Delta t_k)^{m+1} \leq C_0 (h_{max})^{m+1}$). And similarly since it is assumed that $\|\mathbf{Z}_0 - \mathbf{Z}(t_0)\| \approx O(h_{max})^{m+1}$, there exists a constant C_z such that $\|\mathbf{Z}_0 - \mathbf{Z}(t_0)\| \leq C_z (h_{max})^{m+1}$. Therefore the *global error* is,

$$\|\mathbf{Z}_{k+1} - \mathbf{Z}(t_{k+1})\| < \exp((t_{k+1} - t_0) N_{\mathrm{f}} LMC_\beta) \left\{C_z (h_{max})^{m+1} + (k+1) C_0 (h_{max})^{m+1}\right\} . \tag{2.71}$$

Since $\mathrm{T} = t_{k+1} - t_0$, one can easily show that $\frac{T}{h_{max}} \leq k + 1 \leq \frac{T}{h_{min}}$. Using Eq. (2.30) one can show that $k + 1 \leq \frac{TC_t}{h_{max}}$ and substituting the same in the above equation (Eq. 2.71),

$$\|\mathbf{Z}_{k+1} - \mathbf{Z}(t_{k+1})\| < \exp\left(TN_{\mathrm{f}}LMC_\beta\right)\left\{C_z\left(h_{max}\right)^{m+1} + TC_tC_0\left(h_{max}\right)^m\right\}, \tag{2.72}$$

$$< \exp\left(TN_{\mathrm{f}}LMC_\beta\right)\left\{C_z h_{max} + TC_tC_0\right\}\left(h_{max}\right)^m. \tag{2.73}$$

As we are concerned with the behavior of the *global error* as $h_{max} \to 0$, it is safe to assume that there exist a constant C_h such that $h_{max} \leq C_h$, therefore the *global error* can be written of the form,

$$\|\mathbf{Z}_{k+1} - \mathbf{Z}(t_{k+1})\| < C\left(h_{max}\right)^m, \tag{2.74}$$

where C= $\exp\left(TN_{\mathrm{f}}LMC_\beta\right)\left\{C_zC_h + TC_tC_0\right\}$. Hence proved. □ □

Essentially, *Theorem* 3 proves that if $\mathbf{F}^{[j]}$ are sufficiently differentiable, then the solution of MASM not only converges similar to its corresponding *synchronous* solution but also shares the same order of convergence.

Even though the convergence of Adams type MASM method is shown in above theorems, it can be extended to show convergence of non-Adams type MASM solution where $\mathbf{A}$ (2.40), is a function of α_i (2.2), using above theorems and *Lemma* 4.4 [28, Section III.4].

2.4 Numerical Studies and Discussion

In this section we present results for the MASM integrators applied to two test systems, a single-degree-of-freedom linear spring-mass system and a high-dimensional nonlinear aerosol dynamics problem. To compare the MASM results to those of *synchronous* Adams-Bashforth multistep methods, we computed the error versus cost curves for each method.

The error was computed as the difference of the final system state to an analytical

solution or a very accurate numerical solution. Accurate numerical solutions is computed with a high-order adaptive *synchronous* multistep method (the `ode113` routine in `MATLAB`) with very low error tolerances. We used the two-norm $\|\cdot\|_2$ for the spring-mass system and the sup-norm $\|\cdot\|_\infty$ for the aerosol system. The sup-norm was chosen to avoid neglecting the very small aerosol particles, as the key goal of such simulations is to accurately compute the evolution of all particles in the system.

The measure of cost used for comparisons is the number of evaluations of the component functions $\mathbf{F}^{[j]}$ from (2.4). This is an appropriate measure of cost if the function evaluations dominate the other simulation costs, such as solving for the $\left\langle \boldsymbol{\beta}_k^{[j]} \right\rangle_i$ coefficients (2.29). As discussed in Section 2.4.2, this is a valid assumption as the full aerosol condensation equations are complicated implicit functions of the rates, so simply evaluating $\mathbf{F}^{[j]}$ typically involves a Newton iteration. This cost model, however, is not reasonable for simple linear systems, such as the spring-mass problem of Section 2.4.1.

2.4.1 Linear mass-spring system

Consider a single-degree-of-freedom linear spring-mass system, as shown in Figure 2.2. The governing equation is

$$M\ddot{x} + (K_1 + K_2)x = 0, \tag{2.75}$$

which we split as

$$\dot{\mathbf{X}} = \begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1/2 \\ -K_1/M & 0 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix}}_{\mathbf{F}^{[1]}(\mathbf{X})} + \underbrace{\begin{bmatrix} 0 & 1/2 \\ -K_2/M & 0 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix}}_{\mathbf{F}^{[2]}(\mathbf{X})}. \tag{2.76}$$

We took initial conditions $x(0) = 0$ and $v(0) = 1$, parameters $M = 2$, $K_1 = 1$, and

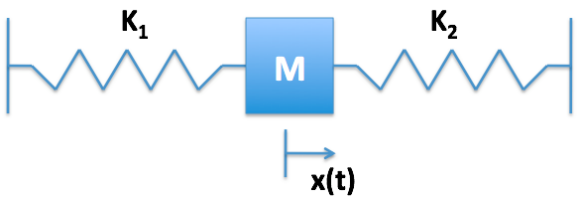

Figure 2.2: Spring-mass system used as a test-case in Section 2.4.1.

$K_2 = 100$, and we chose the timestep for each component to be

$$\Delta t_0^{[p]} = \frac{2\pi h}{10}\sqrt{\frac{M}{K_p}}, \tag{2.77}$$

in terms of a parameter h.

To understand how the performance of the MASM integrators varies with different types of systems, we assume that the costs $\delta^{[1]}$ and $\delta^{[2]}$ of evaluating the functions $\mathbf{F}^{[1]}$ and $\mathbf{F}^{[2]}$ are not equal, and that

$$\frac{\delta^{[2]}}{\delta^{[1]}} = \gamma \tag{2.78}$$

for a constant parameter γ. During a simulation with $N^{[j]}$ evaluations of function $\mathbf{F}^{[j]}$, each at cost $\delta^{[j]}$, the total cost is thus

$$\delta_{\text{total}} = N^{[1]}\delta^{[1]} + N^{[2]}\delta^{[2]} \tag{2.79}$$

$$= \frac{N^{[1]} + \gamma N^{[2]}}{1+\gamma}\delta, \tag{2.80}$$

where $\delta = \delta^{[1]} + \delta^{[2]}$ is the cost of evaluating the total vector field $\mathbf{F}$ (2.4).

Figure 2.3 shows the two-norm error of the system state at time $t = 0.1$ *seconds*, versus cost δ_{total} normalized by δ for different choices of the cost ratio γ, with the curves showing different choices of timestep scale h. Table 2.1 shows the slopes of the curves.

From Figure 2.3 and Table 2.1 we see that the m-step MASM integrators are indeed of order m, as predicted by *Theorem* 1, and that they do not appear to exhibit instabilities for

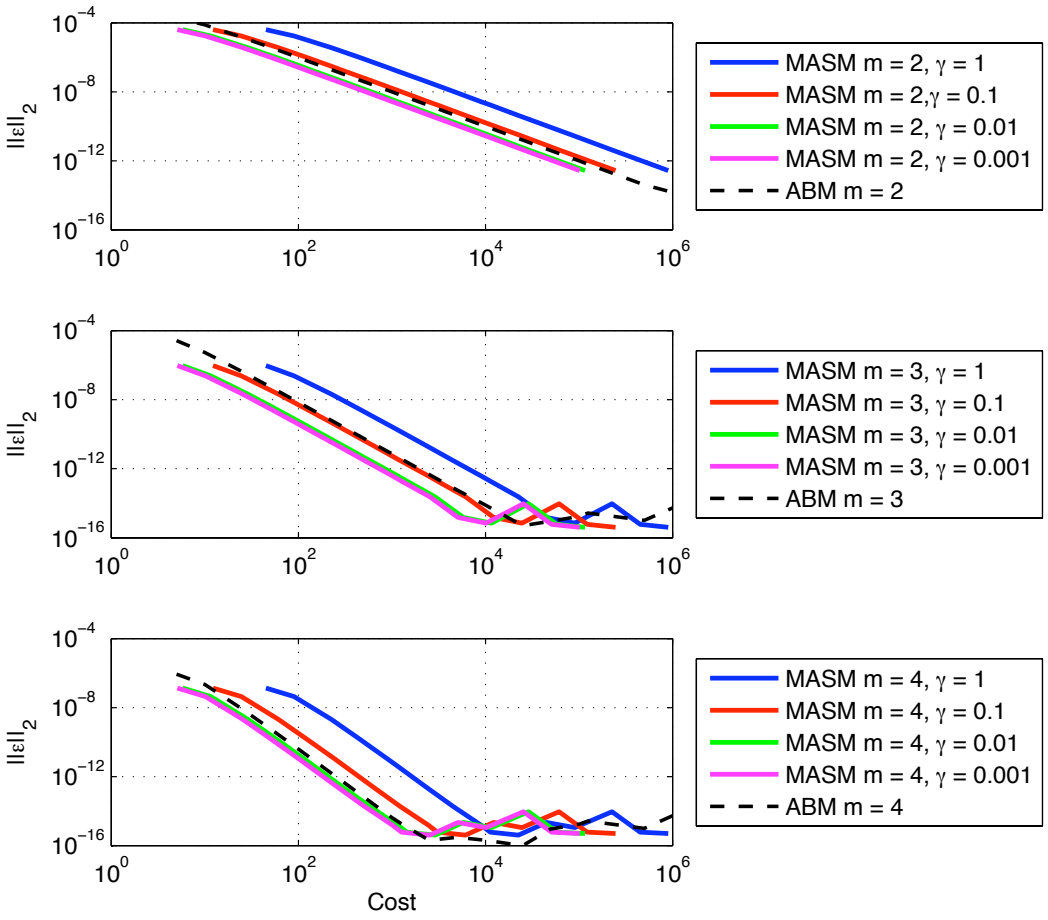

Figure 2.3: Error-versus-cost curves for the linear spring-mass system of Section 2.4.1 using asynchronous MASM and *synchronous* ABM integrators, with varying timestep scale h and with different cost scalings γ (2.78). The cost is $\delta_{\text{total}}/\delta$ (2.80), while the error is the two-norm state error at the final time. We observe that the m-step methods are of order m in both the *synchronous* and asynchronous cases. The asynchronous methods do not offer any improvement in efficiency except for very extreme values of γ. See Table 2.1 for the slopes of each curve.

Table 2.1: Slopes of error-versus-cost curves in Figure 2.3 for the spring-mass system.

	$m = 2$	$m = 3$	$m = 4$
MASM, $\gamma = 1$	-1.97	-2.97	-3.84
MASM, $\gamma = 0.1$	-1.99	-2.95	-3.86
MASM, $\gamma = 0.01$	-1.99	-2.97	-3.85
MASM, $\gamma = 0.001$	-2.00	-2.95	-3.89
ABM (synch)	-2.00	-2.96	-3.95

Table 2.2: Slopes of error-versus-cost curves in Figs. (2.4, 2.5) for the spring-mass system.

	$m=2$	$m=3$	$m=4$	$m=5$
MASM, $\gamma=1$	-1.97	-2.97	-3.84	-4.84
MASM, $\gamma=0.01$	-1.99	-2.95	-3.86	-4.86
ABM (synch)	-2.00	-2.96	-3.95	-4.95

this system. However, the cost of the MASM integrators is almost always higher than the corresponding *synchronous* ABM integrator, except for very extreme values of the cost ratio γ which make the fast component $\mathbf{F}^{[2]}$ much cheaper than the slow component $\mathbf{F}^{[1]}$.

This result is not surprising, as the two system components are very strongly coupled in the system, and so there is little gain from computing one component much less accurately than the other, unless the cost ratios are very skewed.

To explore the effect of coupling between different system components, we look at the numerical performance from modeling (2.75), using a different force field splitting, which is given by,

$$\dot{\mathbf{X}} = \begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ -K_1/M & 0 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix}}_{\mathbf{F}^{[1]}(\mathbf{X})} + \underbrace{\begin{bmatrix} 0 & 0 \\ -K_2/M & 0 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix}}_{\mathbf{F}^{[2]}(\mathbf{X})}. \tag{2.81}$$

As can be seen the, the only difference when compared to the splitting employed earlier (2.76) is that the momentum as a whole is updated using the smaller timestep. Simulations are carried out using the same initial conditions and other parameter values including the timestep (2.77) for each system component. Convergence of the two-norm global error in the numerical solution obtained using different order MASM (*i.e* $m = 2, 3, 4, 5$) against the scaling factor h are shown in Figs. (2.4,2.5). Figure 2.4 shows the numerical convergence with cost scaling $\gamma = 1$, while the figure 2.5 shows the same for $\gamma = .01$. These plots also show a comparison with the corresponding *synchronous* methods. Table 2.2 shows the slope of the curves.

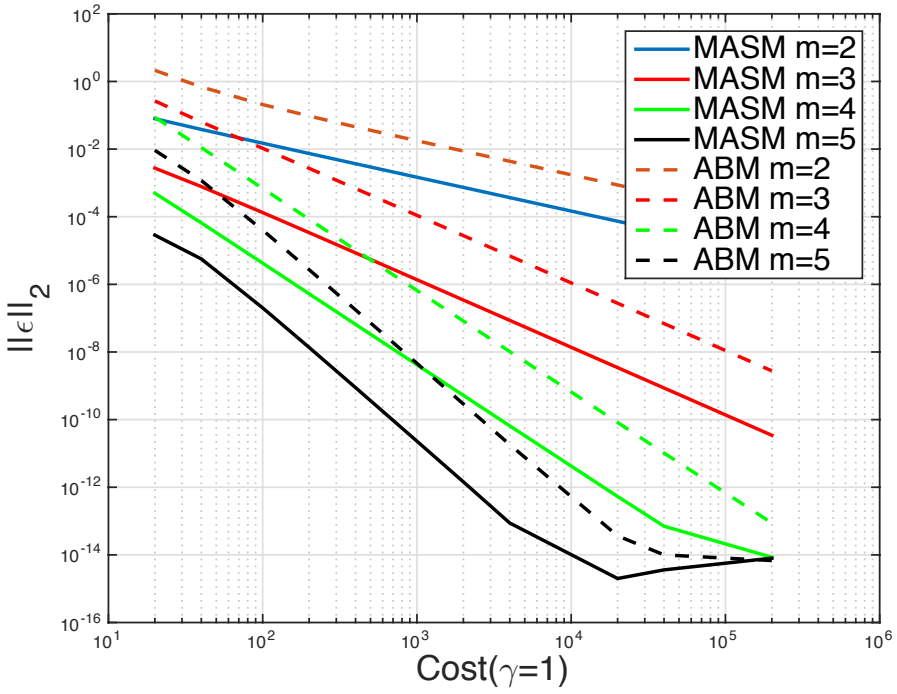

Figure 2.4: Error-versus-cost curves for the linear spring-mass system of Section 2.4.1 using asynchronous MASM with force field splitting (2.81) and *synchronous* ABM integrators, with varying timestep scale h. The cost is $\delta_{\text{total}}/\delta$ (2.80), while the error is the two-norm state error at the final time. We observe that the m-step methods are of order m in both the *synchronous* and asynchronous cases. The asynchronous methods show considerable improvement in efficiency which can be as high as an order when cost ratio $\gamma = 1$ (2.78). See Table 2.2 for the slopes of each curve.

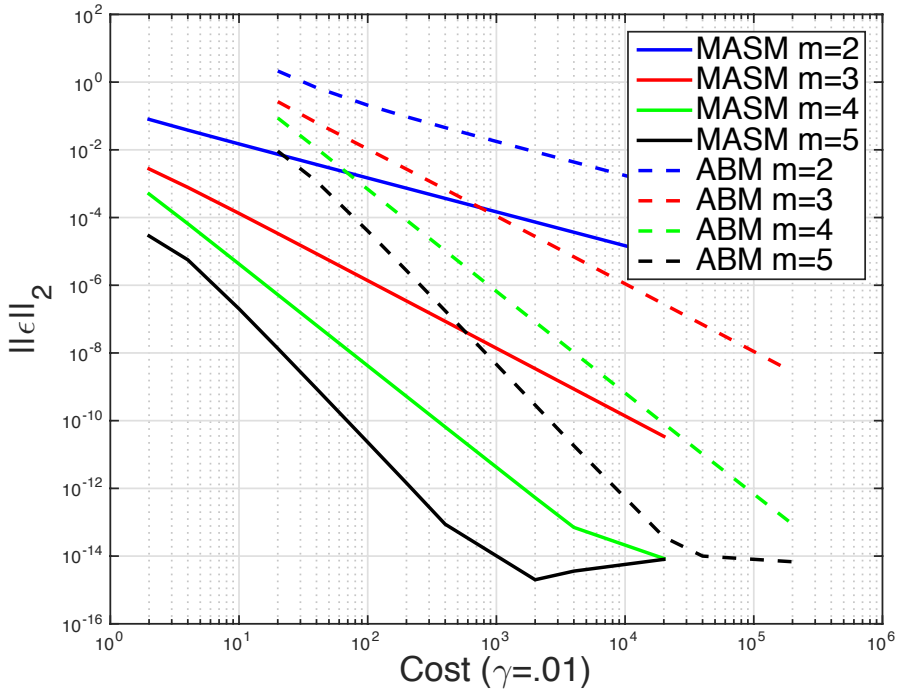

Figure 2.5: Error-versus-cost curves for the linear spring-mass system of Section 2.4.1 using asynchronous MASM with force field splitting (2.81) and *synchronous* ABM integrators, with varying timestep scale h. The cost is $\delta_{\text{total}}/\delta$ (2.80), while the error is the two-norm state error at the final time. We observe that the m-step methods are of order m in both the *synchronous* and asynchronous cases. The asynchronous methods show considerable improvement in efficiency which can be as high as two-orders when cost ratio $\gamma = .01$ (2.78). See Table 2.2 for the slopes of each curve.

From Figure (2.4, 2.5) and Table 2.2, it can be clearly seen that, the numerical solution using MASM is considerably more efficient than its *synchronous* counterparts. The efficiency gain can be as high as *two*-orders for smaller values of cost ratio γ. This result clearly indicates the importance of the coupling between system components. We will explore this aspect in detail, later in Chapter 4.

2.4.2 Nonlinear aerosol condensation system

We now consider a simplified model of water condensation onto aerosol particles. The full dynamics of this process are complicated (see, e.g., Seinfeld and Pandis [55, Chapter 17]), so for simplicity of exposition we use an extremely simple model here. The cost of evaluating the vector field components $\mathbf{F}^{[j]}$ for the full dynamics is very high, so measuring cost as the total number of component evaluations is a reasonable measure for this system.

We take n aerosol particles with volumes $V_1, \ldots, V_n$ in a parcel of air containing water vapor with equivalent liquid-water volume W. We assume that water condensation onto the aerosol particles is proportional to their surface areas and proportional to the water vapor volume-equivalent. This simplified model neglects many key physical processes (e.g. equilibrium vapor state, diffusion limits, curvature terms, etc) and is not at all accurate. It does, however, capture the numerically important features of the full system and so represents a good test case. The system dynamics are

$$\dot{V}_p = W V_p^{\frac{2}{3}} \text{ for } p = 1, 2, .., n \tag{2.82}$$

$$\dot{W} = -\sum_{p=1}^{n} \dot{V}_p = -W \sum_{p=1}^{n} V_p^{\frac{2}{3}}. \tag{2.83}$$

The key feature of this system is that the evolution of the various particles is only sparsely coupled via the global W state, and that the relative growth rate $\dot{V}_p/V_p$ of particles is inversely proportional to their size, so small particles grow relatively quickly.

We split the system into the $n+1$ components given by (2.82) and (2.83) and choose

per-component timesteps

$$\Delta t_0^{[p]} = h\frac{V_p}{|\dot{V}_p|} \text{ for } p = 1, \ldots, n \tag{2.84}$$

$$\Delta t_0^{[n+1]} = h\frac{W}{|\dot{W}|}, \tag{2.85}$$

where h is a scaling parameter.

The aerosol particle volumes were initially normally distributed, with a total of $n = 72$ particles having volumes V_p given by

$$V_p = r_{i+1}U_p + r_i(1 - U_p) \text{ for } n_{i-1} < p \leq n_i \tag{2.86}$$

$$r_i = 10^{-1}\frac{i}{19} + 10^{-8}\left(1 - \frac{i}{19}\right) \text{ for } i = 0, \ldots, 19 \tag{2.87}$$

$$n_i - n_{i-1} = [5\exp(-(r_i - 0.1)^2/0.009)] \text{ for } i = 1, \ldots, 19 \text{ and } n_0 = 0 \tag{2.88}$$

$$U_p \sim \text{Unif}(0, 1), \tag{2.89}$$

where $[x]$ is the nearest integer to x, and $W(0) = 20$. The initial distribution had the largest particle, about 1800 times the volume of the smallest particle, and the ratio of largest to smallest initial timesteps was about 160.

Figure 2.6 shows the sup-norm error of the system state at time $t = 1/10$ versus the cost (total number of component function evaluations). As discussed above, the sup-norm was chosen to ensure that the error in the small particles was not completely dominated by the error in the large particles, as in practical simulations it is important that all particles are accurately simulated (indeed, often it is the small particles which are of most interest). The timestep scaling h was varied to trace out the cost-error curves, and Table 2.3 shows the slopes of these curves.

From Figure 2.6 and Table 2.3 we see that the m-step MASM integrators are again converging at order m, as we expect from *Theorem* 1, and that no instability is apparent.

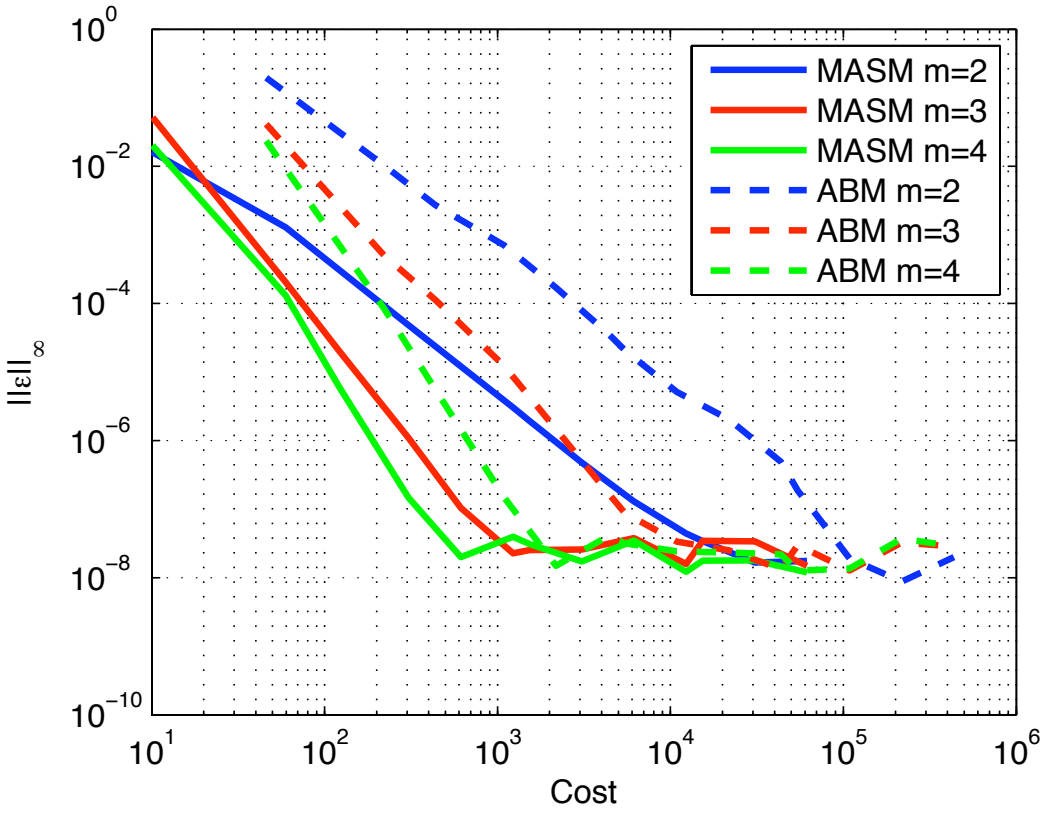

Figure 2.6: Error-versus-cost curves for the nonlinear aerosol condensation system of Section 2.4.2 using the asynchronous MASM and *synchronous* ABM integrators, with varying timestep scale h. The cost is proportional to the total number of component function evaluations, while the error is the sup-norm state error at the final time. We see that the m-step methods are of order m, as expected from *Theorem* 1, and that for this sparsely-coupled system the MASM are more efficient than ABM. See Table 2.3 for the slopes of each curve.

Table 2.3: Slopes of cost-versus-error curves in Figure 2.6 for the aerosol system.

	$m = 2$	$m = 3$	$m = 4$
MASM	-1.98	-3.20	-4.10
ABM	-1.95	-2.70	-3.84

In contrast to the tightly-coupled spring-mass system in Section 2.4.1, here we see that the MASM integrators are more efficient than the equivalent *synchronous* methods. This is due to the wide range of timescales in this problem, and the fact that the different particles are only sparsely coupled.

Considering the implementation cost alone, MASM methods are particularly efficient for problems such as this aerosol simulation, as the component function evaluations $\mathbf{F}^{[j]}$ need to have only the non-zero components stored, and they are almost entirely zero.

Algorithm 1 MASM algorithm (m-step explicit Adams type)

Input: final time $t_{\rm f}$, initial component times $t^{[j]}_{-i}$ and states $\mathbf{\Phi}_{-i}$ for $i = 0, \ldots, m-1$ and $j \in \{1, \ldots, N_{\rm f}\}$

Initialize: $k \leftarrow 0$, $\ell^{[j]}(k) \leftarrow 0$ for $j = 1, \ldots, N_{\rm f}$ and $t_k \leftarrow \max_j t^{[j]}_{\ell^{[j]}(k)}$, with $\mathbf{\Phi}_k$ the corresponding state

Initialize: choose per-component timesteps $\Delta t^{[j]}_{\ell^{[j]}(0)}$

Initialize: $\mathbf{F}^{[j]}_{\ell^{[j]}(0)-i} \leftarrow \mathbf{F}^{[j]}(\mathbf{\Phi}_{\ell^{[j]}(0)-i}, t^{[j]}_{\ell^{[j]}(0)-i})$ for $i = 0, \ldots, m-1$ and $j = 1, \ldots, N_{\rm f}$

while $t_k < t_{\rm f}$ **do**

  $\Delta t_k \leftarrow \min_j \left(t^{[j]}_{\ell^{[j]}(k)} + \Delta t^{[j]}_{\ell^{[j]}(k)} - t_k \right)$

  $t_{k+1} \leftarrow t_k + \Delta t_k$

  for $j = 1$ to $N_{\rm f}$ **do**

    $\tau^{[j]}_{\ell^{[j]}(k)-i} \leftarrow (t^{[j]}_{\ell^{[j]}(k)-i} - t_k)/\Delta t_k$ for $i = 0, \ldots, m-1$

    $\left[\mathbf{V}^{[j]}_k\right]_{p+1,i+1} \leftarrow \left(\tau^{[j]}_{\ell^{[j]}(k)-i}\right)^p$ for i, $p = 0, \ldots, m-1$

    $\langle \mathbf{R} \rangle_{p+1} = \frac{1}{p+1}$ for $p = 0, \ldots, m-1$

    Solve $\mathbf{V}^{[j]}_k \, \boldsymbol{\beta}^{[j]}_k = \mathbf{R}$ for the vector $\boldsymbol{\beta}^{[j]}_k$

  end for

  $\mathbf{\Phi}_{k+1} \leftarrow \mathbf{\Phi}_k + \Delta t_k \sum_{i=0}^{m-1} \sum_{j=1}^{N_{\rm f}} \left\langle \boldsymbol{\beta}^{[j]}_k \right\rangle_i \mathbf{F}^{[j]}_{\ell^{[j]}(k)-i}$

  for $j = 1$ to $N_{\rm f}$ **do**

    if $t_{k+1} = t^{[j]}_{\ell^{[j]}(k)} + \Delta t^{[j]}_{\ell^{[j]}(k)}$ **then**

      $t^{[j]}_{\ell^{[j]}(k+1)} \leftarrow t_{k+1}$, $\mathbf{F}^{[j]}_{\ell^{[j]}(k+1)} \leftarrow \mathbf{F}^{[j]}(\mathbf{\Phi}_{k+1}, t_{k+1})$

      $t^{[j]}_{\ell^{[j]}(k+1)-i} \leftarrow t^{[j]}_{\ell^{[j]}(k)-(i-1)}$, $\mathbf{F}^{[j]}_{\ell^{[j]}(k+1)-i} \leftarrow \mathbf{F}^{[j]}_{\ell^{[j]}(k)-(i-1)}$ for $i = 1, \ldots, m-1$

      update timestep $\Delta t^{[j]}_{\ell^{[j]}(k+1)} > 0$

    else

      $t^{[j]}_{\ell^{[j]}(k+1)-i} \leftarrow t^{[j]}_{\ell^{[j]}(k)-i}$, $\mathbf{F}^{[j]}_{j,\ell^{[j]}(k+1)-i} \leftarrow \mathbf{F}^{[j]}_{j,\ell^{[j]}(k)-i}$ for $i = 0, \ldots, m-1$

      if desired, choose new timestep $\Delta t^{[j]}_{\ell^{[j]}(k+1)} > t_{k+1} - t^{[j]}_{\ell^{[j]}(k)}$

    end if

  end for

  $k \leftarrow k + 1$

end while

Finalize: compute final state $\mathbf{\Phi}_{\rm f}$ at time $t_{\rm f}$ with an mth order interpolant.

Chapter 3

Stability of MASM

The performance of methods addressing multiple-time-scale problems is often limited due to issues related to stability. In this chapter, we address the stability of MASM [27]. We show that MASM is conditionally stable and we study this conditionality in the case of different force field splittings both analytically and numerically. The stability of numerical solutions obtained using MASM for a choice of timesteps strongly depends upon the splitting of the force fields. In a favorable splitting, the computational gain achieved is equivalent to complete modal decomposition.

The outline of the chapter is as follows. Section 3.1 presents the analytical findings addressing the linear stability of MASM. Section 3.2 then investigates the linear stability of MASM via numerical experiments.

3.1 Stability

Similar to many non-symplectic methods, linear stability of MASM is necessary and sufficient for numerical stability [54].

Linear stability of MASM can be studied using *Dahlquist I* (also called *Zero-Stability*) and *Dahlquist II* criteria. *Dahlquist I* looks at the boundedness of the error in solution as the timesteps approach zero *i.e.*, $\Delta t \to 0$. It can be clearly seen that *Adams*-type MASM (2.11), when $\Delta t_k = 0$, is similar to *Euler* method and satisfy *Dahlquist I* criterion unconditionally. According to *Dahlquist II* criteria, a numerical method is stable if there exists values of Δt such that $|\lambda| \leq 1$ (equal only when λ is simple), when applied to an ODE of the form

$\dot{\mathbf{X}}(t) = \lambda \mathbf{X}(t)$ ($\mathbf{X} \in \mathbb{R}^n$), where λ is a scalar constant ($\lambda \in \mathbb{C}$). Application of *Dahlquist II* criteria to MASM would not be simple, given that we are dealing with more than one timesteps. However, we will do so later in the section with certain assumptions and for the rest we will explore the same numerically.

Stability of variable grid multistep methods has been studied in detail in [20, 70, 25, 11]. Adams type multistep methods are also shown to be unconditionally stable on variable grid [20, 70]. These stability studies are performed following the definition of stability by Luxemberg *et. al.* [36]. Following the definition of stability by Luxemberg [36], stability of MASM is proven in the following theorem;

Theorem 4. *When Adams-type MASM (2.11) modeling additively-split ODE system (2.4) with timesteps* $\{\Delta t^{[j]}_{\ell^{[j]}(k)}\}$ *for updating split force vectors, satisfy Eq. (2.30), then the numerical solution obtained is stable.*

Proof. Rewriting Eq. (2.11),

$$\mathbf{\Phi}_{k+1} = \mathbf{\Phi}_k + \Delta t_k \sum_{j=1}^{N_\mathrm{f}} \sum_{i=0}^{M_k-1} \langle \tilde{\boldsymbol{\beta}}^{[j]}_k \rangle_i \left(\mathbf{F}^{[j]}(\mathbf{\Phi}_{k-i}, t_{k-i}) - \mathbf{F}^{[j]}(\mathbf{0}, t_{k-i}) \right) + \Delta t_k \boldsymbol{\lambda}_{k+1} \tag{3.1}$$

where,

$$\boldsymbol{\lambda}_{k+1} = \sum_{j=1}^{N_\mathrm{f}} \sum_{i=0}^{M_k-1} \langle \tilde{\boldsymbol{\beta}}^{[j]}_k \rangle_i \mathbf{F}^{[j]}(\mathbf{0}, t_{k-i}). \tag{3.2}$$

Using intermediate value theorem,

$$\mathbf{\Phi}_{k+1} = \mathbf{\Phi}_k + \Delta t_k \sum_{j=1}^{N_\mathrm{f}} \sum_{i=0}^{M_k-1} \langle \tilde{\boldsymbol{\beta}}^{[j]}_k \rangle_i \mathbf{f}^{[j]}_{\mathbf{\Phi}}(\boldsymbol{\eta}_{k-i}, t_{k-i}) \mathbf{\Phi}_{k-i} + \Delta t_k \boldsymbol{\lambda}_{k+1}, \tag{3.3}$$

where $\boldsymbol{\eta}_{k-i}$ is an intermediate point between $\mathbf{0}$ and $\mathbf{\Phi}_{k-i}$, $\mathbf{f}^{[j]}_{\mathbf{\Phi}} = \frac{d\mathbf{F}^{[j]}}{d\mathbf{\Phi}}$. The above equation can be rewritten as,

$$\mathbf{\Phi}_{k+1} = \mathbf{\Phi}_k + \Delta t_k \sum_{i=0}^{M_k-1} \hat{\beta}_{k,i} \mathbf{\Phi}_{k-i} + \Delta t_k \boldsymbol{\lambda}_{k+1}, \tag{3.4}$$

where,

$$\hat{\beta}_{k,i} = \sum_{j=1}^{N_\mathrm{f}} \langle \tilde{\boldsymbol{\beta}}_k^{[j]} \rangle_i \mathbf{f}_{\mathbf{\Phi}}^{[j]}(\boldsymbol{\eta}_{k-i}, t_{k-i}) \tag{3.5}$$

is a scalar constant. Since timesteps $\{\Delta t^{[j]}_{\ell^{[j]}(k)}\}$ satisfy Eq. (2.30), invoking *Lemma* 2 and *Lemma* 1 we can reduce Eq. (3.2),

$$|\boldsymbol{\lambda}_k| \leq \sum_{j=1}^{N_\mathrm{f}} \sum_{i=0}^{M_k-1} |\langle \tilde{\boldsymbol{\beta}}_k^{[j]} \rangle_i| \max_{i,j} |\mathbf{F}^{[j]}(\mathbf{0}, t_i)|, \tag{3.6}$$

$$\leq N_\mathrm{f} M C_\beta \max_{i,j} |\mathbf{F}^{[j]}(\mathbf{0}, t_i)|. \tag{3.7}$$

where $M_k \leq M = (m+1) N_\mathrm{f} C_t$. Similarly invoking *Lemma* 2 and *Lemma* 1, we can reduce Eq. (3.5) to

$$\sum_{i=0}^{M_k-1} |\hat{\beta}_{k,i}| \leq M N_\mathrm{f} C_\beta L, \tag{3.8}$$

where the fact that $|\mathbf{f}_{\mathbf{\Phi}}^{[j]}| \leq L$ (L is the lipschitz constant of $\mathbf{F}$) is used. Using Eqs. (3.7, 3.8) one can invoke *Theorem* 5 to give an upper bound to the numerical solution, given by

$$\mathbf{\Phi}_{k+1} \leq e^{N_\mathrm{f} M C_\beta L (t_k - t_{M-1})} \left[\Phi + N_\mathrm{f} M C_\beta \max_{i,j} |\mathbf{F}^{[j]}(\mathbf{0}, t_i)| \, (t_k - t_{M-1}) \right], \tag{3.9}$$

where, $\Phi = \max\{\mathbf{\Phi}_0, \mathbf{\Phi}_1, \ldots, \mathbf{\Phi}_{M-1}\}$. Hence the numerical solution obtained using *Adams*-type MASM is stable.

□

The above *Theorem* proves that there exist timesteps such that numerical solution obtained using MASM is stable in Luxumberg [36] sense. We will quantify the stability region,

where the timesteps belonging to the stability region result in a stable numerical solution, analytically in this section and numerically in the next section. To do so, we classify the different kinds of force field splittings (2.4), which play a very important role.

Definition 4 (Asynchronous Splitting)**.** *Consider a linear ODE system with force vector field splitting of the form,*

$$\dot{\mathbf{X}}(t) = \mathbf{F}(\mathbf{X}, t) = \mathbf{A}\mathbf{X}(t) = \sum_{j=1}^{N_{\mathrm{f}}} \mathbf{B}^{[j]}\mathbf{X}. \tag{3.10}$$

where $\mathbf{A}$ *is diagonalizable. If* $(\lambda_i, \boldsymbol{\nu}_i)$ *are the eigen pairs of* $\mathbf{A}$ *and*

$$\mathbf{A}^{[j]} = \lambda_j \left(\boldsymbol{\nu}_j \otimes \boldsymbol{\nu}_j\right), \tag{3.11}$$

then the force field splitting in (3.10) is called Asynchronous Splitting *when* $\alpha_{j,i} \in \{0, 1\}$*, where*

$$\mathbf{B}^{[j]} = \sum_i \alpha_{j,i}\mathbf{A}^{[i]}. \tag{3.12}$$

Asynchronous splitting as defined in the *Definition* 4 is achieved when each eigen component, $\boldsymbol{\nu}_i$, of the system is lumped with one of the split force vectors. When such an idealistic splitting is achieved, each split force field evolves independently. When the number of component force fields that the force field is split into, is equal to Rank($\mathbf{A}$) and satisfy *asynchronous splitting* criteria, it is equivalent to complete *modal decomposition*. Such an idealized splitting is seldom achieved. More often than not, the eigen components are shared among component force vectors. When an eigen component is shared between two or more component force vectors we refer to it as *time scale splitting*.

Application of *Dahlquist II* criteria, when force field splitting employed in MASM satisfy *asynchronous splitting* criteria, is very straight forward. In the case of *time scale splitting*, application of *Dahlquist II* criteria is studied numerically in the next section.

Application of *Dahlquist II* criteria is equivalent to the application of Lagrange Method to MASM for linear ODE system 3.10, where $\mathbf{A}$ is diagonalizable and $(\lambda_i, \boldsymbol{\nu}_i)$ are its eigen pairs. Expressing $\mathbf{X}(t) = \sum_i a_i(t)\boldsymbol{\nu}_i$, one can reduce the linear multistep method to system of equations of the form,

$$a_i(t_{k+1}) - a_i(t_k) - \mu \sum_j \beta_j a_i(t_{k-j}) = 0,$$

where $\mu = \Delta t \times \lambda_{max}$. For stability, we solve for $\zeta(\mu)$, where $a_i(t_{k+1}) = \zeta \times a_i(t_k)$, such that

$$\zeta^{n+1} - \zeta^n - \mu \sum_i \beta_i \zeta^{n-i} = 0, \tag{3.13}$$

satisfied. Linear stability region of linear multistep methods can be given by,

$$\mathcal{S} = \{\mu \in \mathbb{C} \,|\text{Root of Eq. (3.13) satisfy}, |\zeta(\mu)| \leq 1\} \tag{3.14}$$

Using a very similar approach, one can reduce application of MASM to ODE system (3.10) which satisfies *asynchronous splitting* criteria, to a system of characteristic equations of the form,

$$\zeta^{n+1} - \zeta^n - \mu^{[j]} \left(\sum_i^{m-1} \beta_i^{[j]} \zeta^{n-i} \right) = 0. \tag{3.15}$$

where $\mu^{[j]} = \Delta t^{[j]} \times \max\{|\lambda_i| \left| \boldsymbol{\nu}_i^T \mathbf{B}^{[j]} \boldsymbol{\nu}_i \neq 0 \right.\}$. Therefore, MASM when modeling an ODE system (3.10), where the forcing field splitting satisfy *asynchronous splitting* criteria, is stable if only if

$$\mu^{[j]} \in \mathcal{S} \text{ (see Eq. 3.14)} \quad \text{for all } j. \tag{3.16}$$

While solving real engineering problems, the force vector splitting seldom satisfy *Asyn-*

chronous Splitting criteria. Often in a force vector splitting, some of the eigen components are split between the component force vectors. As a result any force vector splitting is a combination of certain eigen components being lumped in a single component force vector, while others are split between different component force vectors, which is referred to as *time scale splitting*. We will only explore the stability criteria in the case of 2-component *time scale splitting* in this study.

3.2 Numerical Studies and Discussion

While the linear stability of the MASM and the stability criteria on systems where *asynchronous splitting* is achieved is proven in the previous section, in this section we will numerically study the stability criterion in the case of *time scale splitting*, especially when the eigen component is split between two component force fields.

Stability of the numerical method is studied using the following methodology. Given $\Psi_{\Delta t_k}$: $\mathbb{R}^{M_k \times M} \to \mathbb{R}^M$ is the one-step integrator of MASM applied to differential equations given by Eq. (2.4) whose vector field can be split into several components also given by Eq. (2.4), MASM can be represented by,

$$\mathbf{X}_{k+1} = \Psi_{\Delta t_k} \left(\{t_{1,k-i}\}_{i=0}^{m-1}, \ldots, \{t_{N,k-i}\}_{i=0}^{m-1} \right) \{\mathbf{X}_{k-i}\}_{i=0}^{M_k-1}. \tag{3.17}$$

In the case of linear systems, such as the examples studied in this section, numerical method can be written as,

$$\mathbf{Z}_{k+1} = \mathbf{A}_\Psi \mathbf{Z}_k, \tag{3.18}$$

where $\mathbf{Z}_k^T = [\mathbf{X}_k^T \ldots \mathbf{X}_{k-(M_k-1)}^T]$ and $\mathbf{A}_\Psi$ is a coefficient matrix. Eigenvalues of $\mathbf{A}_\Psi$ determine the stability of the method *i.e.* the numerical method is stable if $|\lambda_{max}(\mathbf{A}_\Psi)| \leq 1$ (strictly < 1 when *algebraic multiplicity* of $\lambda_{max}(\mathbf{A}_\Psi)$ is greater than 1), where $\lambda_{max}(\mathbf{A}_\Psi)$ is

the eigen value of $\mathbf{A}_\Psi$ with maximum absolute value. In the following numerical studies, $\mathbf{A}_\Psi$ is determined for different values of the parameters and its eigenvalues are used to determine the maximum value of timesteps for which the method is stable.

3.2.1 Conservative Systems

The governing equations of a sdof linear spring-mass system is given by,

$$\dot{\mathbf{X}}(t) = \begin{bmatrix} 0 & 1 \\ -k_t & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ v(t) \end{bmatrix}, \tag{3.19}$$

where k_t is the total stiffness, and x, v are the position and velocity variables respectively. With the eigenvalues of the same being $\pm\sqrt{-k_t}$, it is a single time scale system and it represents any conservative eigen component in a multiscale system. In this section we are only studying the case where an eigen component is shared between two component force vectors. The same can be represented by,

$$\dot{\mathbf{X}} = \begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ -k_t & 0 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix}}_{\mathbf{F}=\mathbf{A}\mathbf{X}} = \underbrace{\left(\frac{r_s}{1+r_s}\right) \begin{bmatrix} 0 & 1 \\ -k_t & 0 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix}}_{\mathbf{F}^{[1]}(\mathbf{X})=\mathbf{B}^{[1]}\mathbf{X}} + \underbrace{\left(\frac{1}{1+r_s}\right) \begin{bmatrix} 0 & 1 \\ -k_t & 0 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix}}_{\mathbf{F}^{[2]}(\mathbf{X})=\mathbf{B}^{[2]}\mathbf{X}} \tag{3.20}$$

$$= \underbrace{\alpha \begin{bmatrix} 0 & 1 \\ -k_t & 0 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix}}_{\mathbf{F}^{[1]}(\mathbf{X})=\mathbf{B}^{[1]}\mathbf{X}} + \underbrace{(1-\alpha) \begin{bmatrix} 0 & 1 \\ -k_t & 0 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix}}_{\mathbf{F}^{[2]}(\mathbf{X})=\mathbf{B}^{[2]}\mathbf{X}} \tag{3.21}$$

where $r_s = \|\mathbf{F}^{[1]}\|/\|\mathbf{F}^{[2]}\|$ is the ratio of the norm of component force vector that is updated using the smaller timestep, Δt_s, against the norm of component force vector updated using larger timestep, Δt_ℓ. Henceforth, r_s will be referred as *splitting ratio* in this thesis. Eq. (3.21) shows a different parameterization than in Eq. (3.20), where α represents the fraction of the eigen component updated using the smaller timestep. In this study, we predominantly employ the parameters used in Eq. (3.20) and the reasons for the same will be touched upon later in the discussion.

Applying MASM to the above governing equations (3.20), matrix $\mathbf{A}_\Psi$ in Eq. (3.18) can

be calculated. Note that the matrix,

$$\mathbf{A}_\Psi = \mathbf{A}_\Psi(r_s, k_t, \{\Delta t_k^{[j]}\} \; \forall \; k \geq 0 \text{ and } j) = \mathbf{A}_\Psi(r_s, k_t, \Delta t_\text{s}, \Delta t_\ell), \tag{3.22}$$

is a function of system dynamics, *splitting ratio* (r_s), as well as the component timesteps chosen. Also note that, while $\mathbf{A}_\psi$ explicitly depends upon the quantities inside the parenthesis, it also implicitly depends upon m. In Eq. (3.22), the second equality results from the fact that the component timesteps chosen are assumed to remain constant for simplicity in the present analysis. Eigenvalues of the matrix $\mathbf{A}_\Psi$ determine the stability of the numerical method for this system, *i.e* the method is stable if $|\lambda_{max}(\mathbf{A}_\Psi)| \leq 1$ (strictly < 1 when *algebraic multiplicity* of $\lambda_{max}(\mathbf{A}_\Psi)$ is greater than 1).

In multistep methods, the largest eigenvalue, $\lambda_{max}(\mathbf{A})$ (see Eq. 3.20), of the system times the timestep, *i.e* $\mu = \Delta t \times \lambda_{max}(\mathbf{A})$, determine the stability. μ ($\sim \Delta t / T_{max}(\mathbf{A})$), where $T_{max}(\mathbf{A})$ is the time period of the eigen component with largest eigenvalue, is a measure of how fast the numerical solution is updated compared to the fastest time scale of the system. Similarly, in MASM μ_ℓ ($= \Delta t_\ell \times \lambda(\mathbf{A})$) and μ_s ($= \Delta t_\text{s} \times \lambda(\mathbf{A})$) are the key parameters that determine stability. Variation of maximum stable value of μ_ℓ and μ_s as *splitting ratio*, r_s, is varied or as n is varied (where $n = \Delta t_\ell / \Delta t_\text{s}$ is the ratio of timesteps, which will also be referred as *timestep ratio* in this thesis) are studied. The results from these studies are compared with when two component timesteps are the same *i.e* $n = 1$, which represents the corresponding linear multistep method or its *synchronous* counterpart.

The variation of the maximum stable values of μ_ℓ and μ_s as the *splitting ratio*, r_s, is varied for different values of the *timestep ratio*, n, are shown in Figs. (3.1, 3.2) respectively. From Figs. (3.1, 3.2) when $r_s \ll 1$, μ_s assumes values which are steadily smaller for increasing values of n, while μ_ℓ remains the same for all n (including $n = 1$). This clearly shows that stability is determined by μ_ℓ and the stability limit is same as the stability limit for its *synchronous* counterpart ($n = 1$). This behavior is similar to any traditional *synchronous*

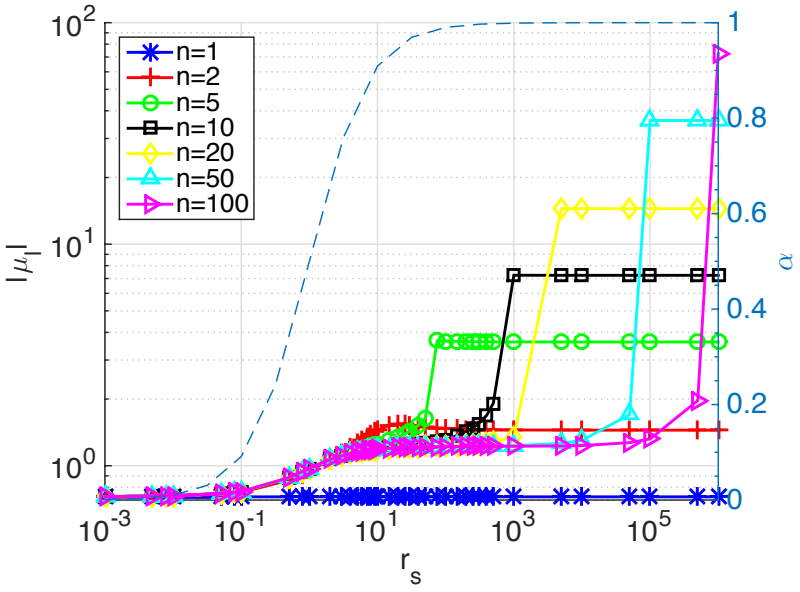

Figure 3.1: This figure shows the variation of maximum stable value of $|\mu_\ell| = \Delta t_\ell \times |\lambda(\mathbf{A})|$ (see Eq. 3.20) as the *splitting ratio*, r_s (see Eq. 3.20), is varied for different values of *timestep ratio* (n), for conservative spring-mass system studied in Section 3.2.1. Variation of α (Eq. 3.21) is also shown. When the fraction of the forcing updated by smaller timestep, Δt_s, is smaller, *i.e* when $r_s \ll 1$, the stability is determined by the larger timestep, Δt_ℓ. The stability *criterion* that μ_s needs to satisfy is same as the stability *criterion* in the *synchronous* case ($n = 1$). As the fraction of the forcing updated by the smaller timestep (Δt_s), increases, μ_ℓ and μ_s undergo transition from larger timestep, Δt_ℓ, determining stability to the smaller timestep, Δt_s, determining stability.

method, where the largest μ determine the stability.

When $r_s \gg 1$, *i.e* when the fraction of the forcing updated by the smaller timestep is significantly larger than the fraction updated by the larger timestep, μ_ℓ and μ_s go through a *transition*. During the *transition*, they transitions from larger timestep determining the stability to smaller timestep determining stability. This *transition* happens at different values of *splitting ratio*, r_s, for different values of the *timestep ratio*, n. For larger values of n, the transition occurs at larger values of r_s and beyond this *transition*, μ_s determine the stability.

Figs. (3.1, 3.2) also show the variation of α with the *timestep ratio*, where one can see that

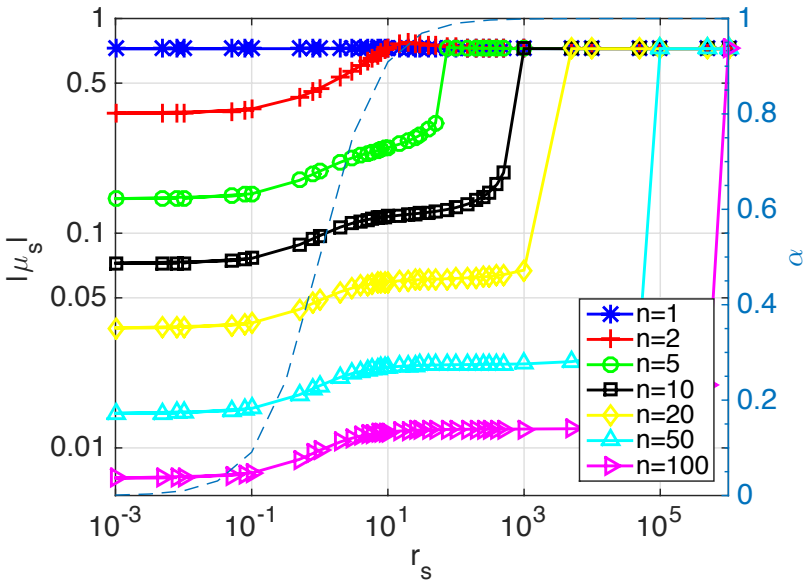

Figure 3.2: This figure shows the variation of maximum stable value of $|\mu_s| = \Delta t_s \times |\lambda(\mathbf{A})|$ (see Eq. 3.20) as the *splitting ratio*, r_s (see Eq. 3.20), is varied for different values of *timestep ratio* (n), for conservative spring-mass system studied in Section 3.2.1. Variation of α (Eq. 3.21) is also shown. The smaller timestep, Δt_s, seems to determine the stability only when larger fraction of the forcing is updated using the smaller timestep, *i.e* when $r_s \gg 1$, especially after μ_s and μ_ℓ undergo transition. The stability *criterion* that μ_s needs to satisfy is same as it is for its *synchronous* counterpart (n=1).

the *transition* happen at much larger values of α (> 0.9). Parameterization in Eq. 3.21 was employed in few studies, but using Eq. 3.20 allowed us to study the dynamics of *transition* better.

Since one can easily vary *timestep ratio* (n) during simulations using timestep adaptation, it is more interesting to see the variation of maximum stable value of μ_ℓ and μ_s as n is varied. The same can be see in Figs. (3.3, 3.4), where the variation of μ_ℓ (Fig. 3.3) and μ_s (Fig. 3.4) are shown as n is varied for different values of r_s. It can be seen that, for values $r_s \gg 1$, μ_ℓ grows while n increases, at the same time μ_s remains constant. This clearly indicates that μ_s determine stability for smaller values of n. As n increases μ_ℓ peaks to a certain value before transitioning to a lower value. This transition is nothing but the *transition* seen in

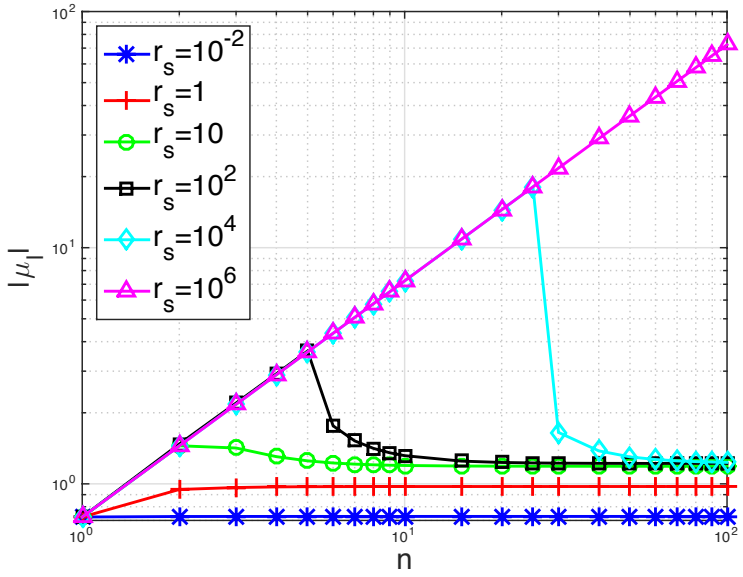

Figure 3.3: This figure shows the variation of maximum stable value of $|\mu_\ell|$ as the *timestep ratio*, n ($= \Delta t_\ell / \Delta t_\mathrm{s}$), is varied for different values of the *splitting ratio*, r_s (see Eq. 3.20), in the case of conservative spring-mass system studied in Section 3.2.1. When the fraction of the forcing that is updated by smaller timestep is large enough, the maximum stable larger timestep, Δt_ℓ, grows with n and peaks before transitioning to a value similar to $n = 1$. When $r_s \ll 1$, Δt_ℓ determines the stability.

Figs. (3.1, 3.2). Beyond this *transition*, one can see that μ_s monotonically reduces, while μ_ℓ remains constant, again indicating that μ_ℓ determines the stability for larger values of n.

The overall behavior of maximum stable values of μ_s and μ_ℓ as *splitting ratio*, r_s, or as *timestep ratio*, n, is varied can be divided into two regions. In *Region I*, Δt_ℓ determines the stability and Δt_s determines stability in *Region II*. These two regions are separated by the *transition* region. As was seen in the Figs. (3.1 - 3.4), the values of *timestep ratio*, n, grow with increasing values of *splitting ratio* in *transition* region. These values of *splitting ratio* and *timestep ratio* in the *transition* region are plotted in Fig. 3.5 for $m = 3$ and $m = 4$. From which one can clearly see that,

$$n \sim r_s^{1/m}, \tag{3.23}$$

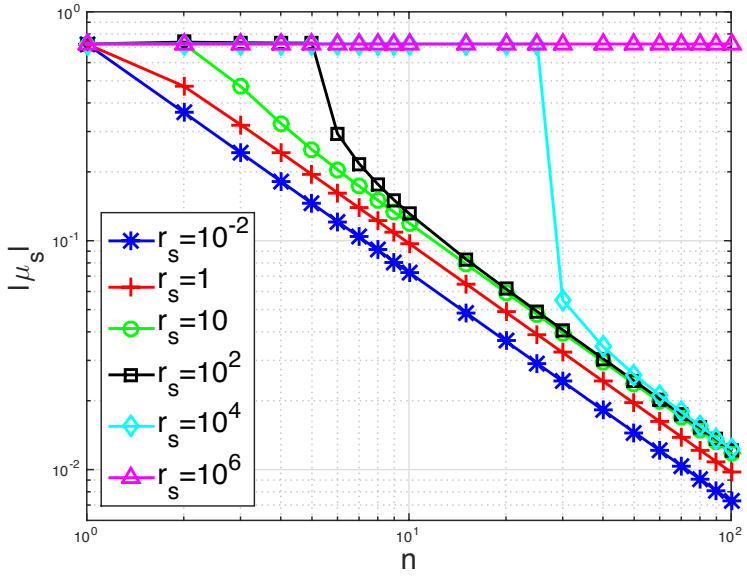

Figure 3.4: This figure shows the variation of maximum stable value of $|\mu_s|$ as the *timestep ratio*, n $(= \Delta t_\ell/\Delta t_s)$, is varied for different values of the *splitting ratio*, r_s, in the case of conservative spring-mass system studied in Section 3.2.1. When the fraction of the forcing that is updated by smaller timestep is large enough, smaller timestep (Δt_s) determine stability before μ_s and μ_ℓ undergo *transition*.

where n and r_s are in *transition* region. One can also observe from Fig. 3.5, when n is large enough and as we move from smaller values of r_s to larger values, we move from *Region I* to *Region II*, which is also observed in Figs. (3.1, 3.2). Similarly, with r_s large enough, we move from *Region II* to *Region I* as we choose larger values of n, also seen in Figs. (3.3, 3.4).

From computational cost point of view, the computational savings compared to its *synchronous* counterpart is minimal when the larger timestep determines stability, *i.e* when in *Region I*. The most optimal choice of timesteps from computational savings point of view when in *Region I*, is when the timesteps are equal, *i.e* $n = 1$, where MASM is equivalent to its *synchronous* counterpart. However, when in *Region II* where the smaller timestep determines the stability allowing the larger timestep to be larger. This results in fewer functions evaluations of the fraction of the force vector that is updated by larger timestep, it leads

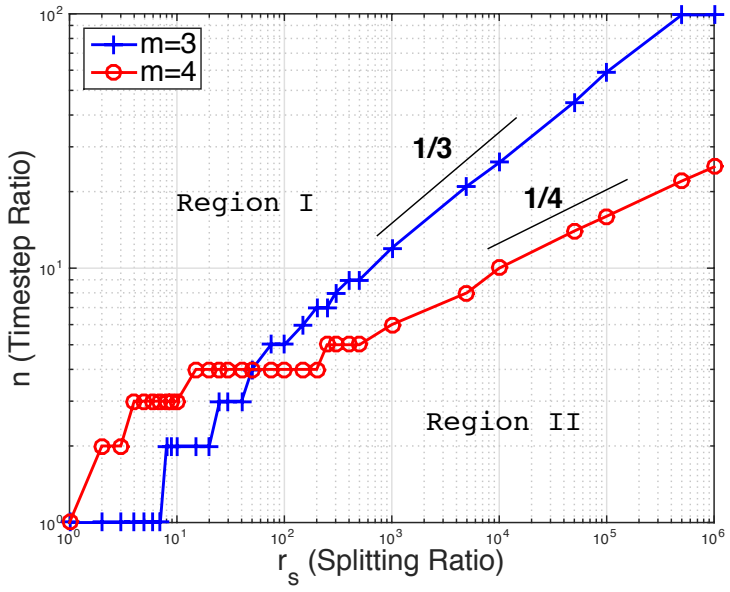

Figure 3.5: Parameter values of *timestep ratio*, n, and the *splitting ratio*, r_s (see Eq. 3.20), at the *transition* are plotted against each other, in the case of conservative spring-mass system studied in Section 3.2.1. *Transition* curve separate *Region I*, where Δt_ℓ determines stability, and *Region II*, where Δt_s determines stability. Maximum computational savings are achieved for the parameter values in the *transition* region. In the *transition* region, *timestep ratio*, n, is proportional to $1/m^{th}$ power of *splitting ratio*, r_s (see Eq. 3.23).

to computational savings. Therefore maximum computational savings are achieved in the *transition* region. Fig. 3.6 shows the computational gain achieved in *transition* region for different values of cost ratio, γ (see Eq. 2.78). Cost ratio curves in Fig. 3.6 clearly show that the computational gain can be significant even in the case of *time scale splitting* with favorable cost ratio (*i.e* $\gamma \ll 1$), when optimum parameter values are chosen (*i.e* in *transition* region).

However, the choice of larger timestep also depends upon other eigen components shared by the respective component force field, $\mathbf{F}^{[2]}$ (see Eq. 3.20). Therefore, when the component force field, $\mathbf{F}^{[1]}$, that is updated by smaller timestep shares a larger fraction of an eigen component, it allows the component force vector ($\mathbf{F}^{[2]}$) that is updated by larger timestep ($\Delta t^{[2]}$) to choose large timestep leading to computational savings.

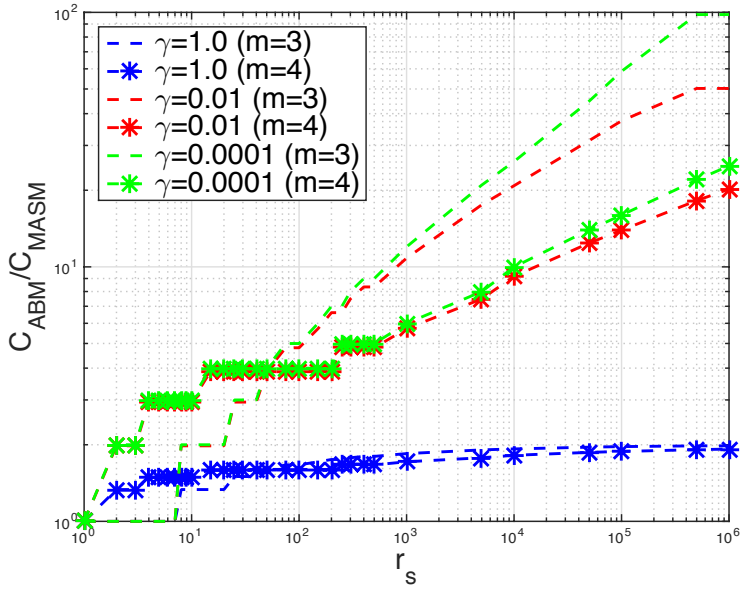

Figure 3.6: Computational cost ratio curves in the case of *time scale splitting* for $m = \{3, 4\}$ and for different values of cost ratio, γ (see Eq. 2.78) are shown for spring-mass system studied in Section 3.2.1. Computational cost using MASM is estimated for the parameter values, $(r_s,\ n)$, in the *transition* region. When the cost of evaluating the fraction of the forcing updated using the smaller timestep is smaller, *i.e* when $\gamma \ll 1$, the efficiency gain that can be achieved using MASM can be of several orders even in the case of *time scale splitting* in conservative systems.

3.2.2 Non-Conservative Systems

In the case of non-conservative systems, we study a sdof spring-mass-damper system whose governing equations are given by,

$$\dot{\mathbf{X}} = \begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ -k_t & -c_t \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix}}_{\mathbf{F}=\mathbf{A}\mathbf{X}} = \underbrace{\left(\frac{r_s}{1+r_s}\right) \begin{bmatrix} 0 & 1 \\ -k_t & -c_t \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix}}_{\mathbf{F}^{[1]}(\mathbf{X})=\mathbf{B}^{[1]}\mathbf{X}} + \underbrace{\left(\frac{1}{1+r_s}\right) \begin{bmatrix} 0 & 1 \\ -k_t & -c_t \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix}}_{\mathbf{F}^{[2]}(\mathbf{X})=\mathbf{B}^{[2]}\mathbf{X}} \tag{3.24}$$

where c_t is the damping in the system. The natural frequencies remain complex conjugate

and represent a single time scale system or an eigen component as long as $c_t \leq 2\sqrt{k_t}$ (*i.e* when the damping ratio $\xi(= c_t/2\sqrt{k_t}) \leq 1$).

Figs. (3.7, 3.8) not only show the variation of maximum stable values of μ_ℓ and μ_s against variation of r_s for different values of n, but also for different values of the damping ratio $\xi = \{\ 0.0,\ 0.05,\ 0.5,\ 1.0\ \}$. It can be clearly seen that as the damping increases, the *transition* is no more dependent upon n. As a result, the timestep that determines the stability is only a function of the fraction of the forcing it updates, *i.e* r_s, when updating eigen modes with reasonable damping. The same can also be seen in Figs. (3.9, 3.10) where with little damping, μ_ℓ either monotonically grows with n or remains constant.

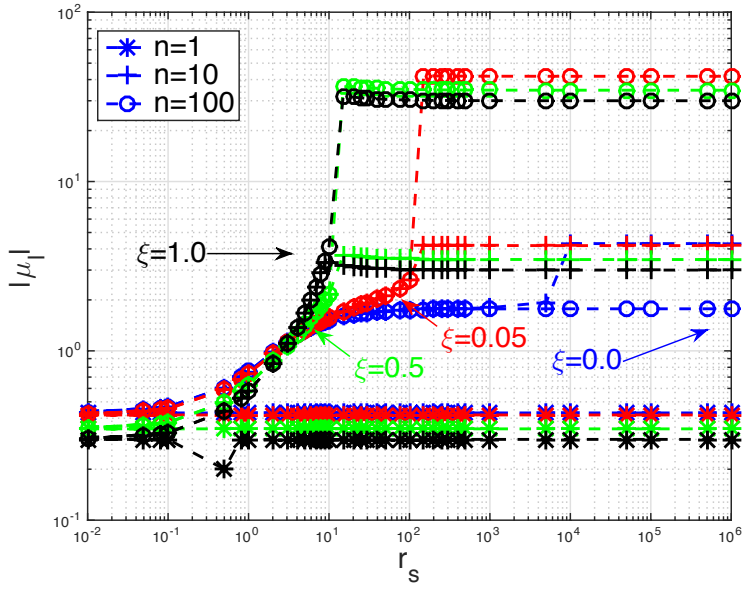

Figure 3.7: This figure shows the variation of maximum stable value of $|\mu_\ell| = \Delta t_\ell \times |\lambda(\mathbf{A})|$ (see Eq. 3.20) as the *splitting ratio*, r_s (see Eq. 3.20), is varied for different values of *timestep ratio* (n) and for different values of *damping ratio*, $\xi = \{0.0, 0.05, 0.5, 1.0\}$, in spring-mass-damper system studied in Section 3.2.1. For systems with damping, the timestep that determines stability is completely determined by the *splitting ratio*, r_s. Smaller fractions of forcing, that is updated using the smaller timestep, are needed with increasing damping in the system at the *transition* point.

For $\xi = \{\ 0.05,\ 0.5,\ 1.0\ \}$, the transition happens at $r_s = \{\ 80,\ 9,\ 8\ \}$. This implies that

when a non-conservative eigen component is shared between two component force vectors, the smaller timestep determines stability when the fraction of the component forcing updated by the smaller timestep is greater than { 98.7%, 90%, 88.8% } for ξ = { 0.05, 0.5, 1.0 } respectively. Therefore, the probability that a splitting could allow an independent choice of larger timestep in non-conservative systems increases with damping. Note that when an independent choice of larger timestep can be made, it potentially could lead to significant computational gains.

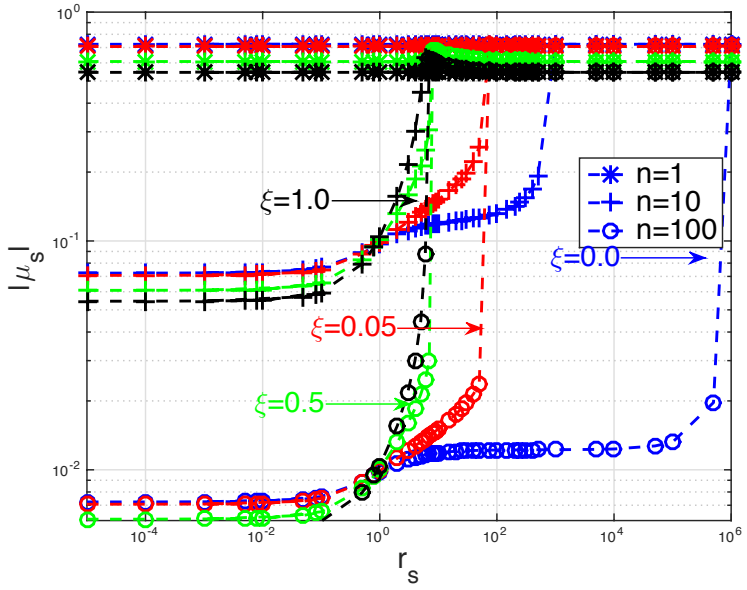

Figure 3.8: This figure shows the variation of maximum stable value of $|\mu_s| = \Delta t_s \times |\lambda(\mathbf{A})|$ (see Eq. 3.20) as the *splitting ratio*, r_s (see Eq. 3.20), is varied for different values of *timestep ratio* (n) and for different values of *damping ratio*, $\xi = \{0.0, 0.05, 0.5, 1.0\}$, in spring-mass-damper system studied in Section 3.2.1. For systems with damping, the timestep that determines stability is completely determined by the *splitting ratio*, r_s. Smaller fractions of forcing that is updated using the smaller timestep are needed with increasing damping in the system, for the smaller timestep to determine stability.

Fig. 3.11 shows the changes in the stability region of μ_s in complex plane as n is varied across the *transition*, (for $r_s = 10^2$) also shown in Fig. 3.1. From this one can clearly see that for larger values of n, the conservative systems and systems with little damping transition

from μ_s determining stability to where μ_ℓ determines the stability. Fig. 3.12 shows the growth of complex stability region of μ_s as the fraction updated by the smaller timestep, r_s, grows for $n = 100$. This figure clearly shows that the complex stability region of μ_s grows with r_s. The figure also show that with higher damping one needs smaller fraction of the forcing to be updated by smaller timestep, such that the smaller timestep determine stability.

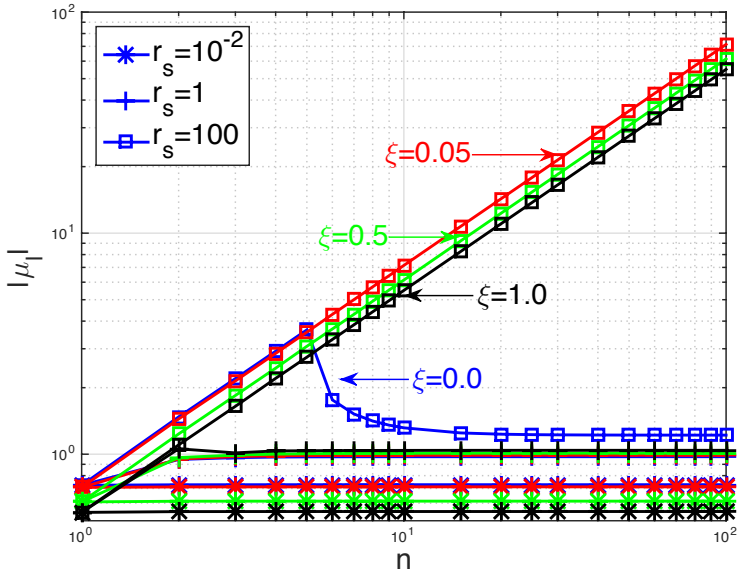

Figure 3.9: This figure shows the variation of maximum stable value of $|\mu_\ell|$ as the *timestep ratio*, n $(= \Delta t_\ell / \Delta t_s)$, is varied for different values of the *splitting ratio*, r_s (see Eq. 3.20), and for different values of *damping ratio*, $\xi = \{0.0, 0.05, 0.5, 1.0\}$, in the spring-mass-damper system studied in Section 3.2.2. With resonable damping in the system and with the fraction of the forcing that is updated by smaller timestep being large enough, the maximum stable value of larger timestep, Δt_ℓ, grows monotonically as *timestep ratio* (n) increases. Indicating that with reasonable damping, the smaller timestep, Δt_s, completely determines the stability.

Until now we have only studied the effect of parameters such as r_s and n, on maximum stable μ_s and μ_ℓ. k_t affects the time scale of the system or the eigen mode via the largest eigen value $\lambda(\mathbf{A})$ which is considered in the parameters $\mu_s = \lambda \times \Delta t_s$ and $\mu_\ell = \lambda \times \Delta t_\ell$. Therefore, when the total stiffness increases, it results in reducing the timesteps uniformly,

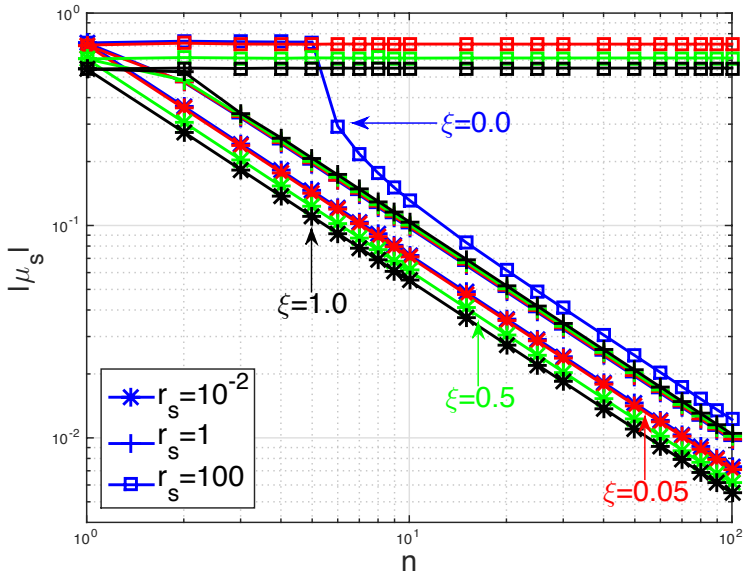

Figure 3.10: This figure shows the variation of maximum stable value of $|\mu_s|$ as the *timestep ratio*, n $(= \Delta t_\ell / \Delta t_s)$, is varied for different values of the *splitting ratio*, r_s (see Eq. 3.20), in the spring-mass-damper system studied in Section 3.2.1. In the case of systems with reasonable damping, the *splitting ratio* (r_s) completely determines which timestep governs stability.

since μ_s and μ_ℓ have to satisfy the same stability *criterion*. Results similar to Figs. 3.1 - 3.12, which were generated using three point forcing history, *i.e* $m = 3$, are also shown for different values of m in Figs. 3.13 - 3.26.

Figs. (3.13, 3.14) show the variation of maximum stable value of μ_s and μ_ℓ as the *splitting ratio*, r_s, is varied for different values of n. Similarly Figs. 3.15, 3.16 show the variation of maximum stable value of μ_s and μ_ℓ as n is varied for different values of *splitting ratio*, r_s. Results in these plots are obtained using MASM which uses four historical force evaulations, *i.e* $m = 4$. From these plots, one can see that when *splitting ratio* is small, *i.e* $r_s \ll 1$, Δt_ℓ determines the stability and for larger values of r_s past the *transition* point, Δt_s determines the stability. It can also be seen that at the *transition* point, potential computational savings are the highest. The relationship between n and r_s at the *transition* point is already shown

in Fig. 3.5 and given by Eq. 3.23. The computational savings that could be achieved at the *transition* point for $m = 4$ is already shown in the Fig. 3.6. These results are very similar to the results we have seen in the case of $m = 3$.

Figs. 3.17-3.20 show the effect of damping on the variation of maximum stable values of μ_s and μ_ℓ as the *splitting ratio* and *timestepratio* is varied, in the case for $m = 4$. These plots show that with reasonable damping, the *splitting ratio* completely determines the timestep that determines the stability, similar to the results in the case of $m = 3$.

Similar to the results in the case of $m = 3$, Figs. (3.21-3.26) show that the changes to the stability region of μ_s in complex plane as n and r_s are varied when $m = 2$, $m = 4$ and $m = 5$. Similar to the case when $m = 3$, Figs. (3.21,3.23,3.25) show that as n is varied, conservative eigen modes as well as the eigen modes with reasonable damping undergo *transition* from μ_s determining stability to where μ_ℓ determines the stability for $m = 2$, $m = 4$ and $m = 5$ respectively. Similarly, Figs. (3.22,3.24,3.26) clearly show that with higher damping, one needs smaller fraction of the forcing to be updated by smaller timestep, such that the smaller timestep determine stability for $m = 2$, $m = 4$ and $m = 5$ respectively.

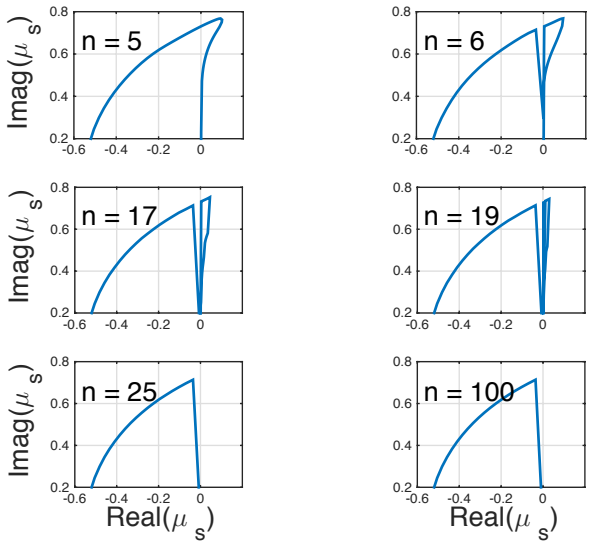

Figure 3.11: Complex stability region of μ_s with different values of *timestep ratio*, n, when $r_s = 100$, are shown. The conservative eigen modes are the first to undergo *transition* and extend to slightly damped eigen modes with increasing the *timestep ratio* (n).

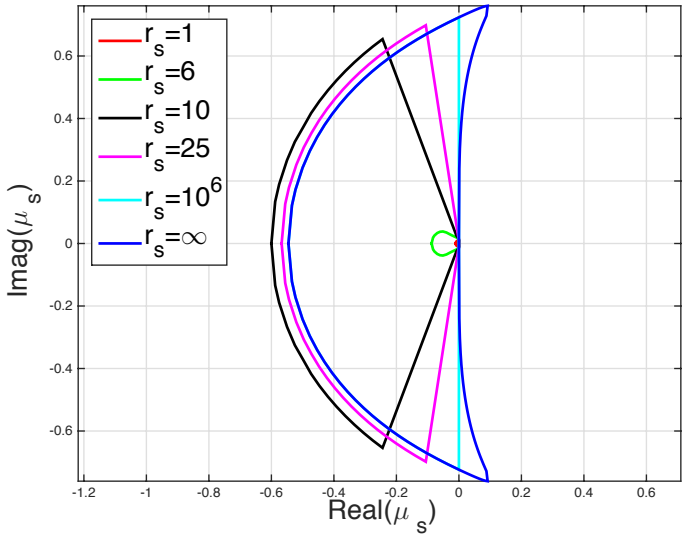

Figure 3.12: Complex stability region of μ_s is shown for different values of the *splitting ratio*, r_s (see Eq. 3.20), and when $n = 100$. Larger the damping ratio is in an eigen mode, smaller is the value of *splitting ratio*, r_s, that is sufficient to allow the choice of larger timestep to be independent.

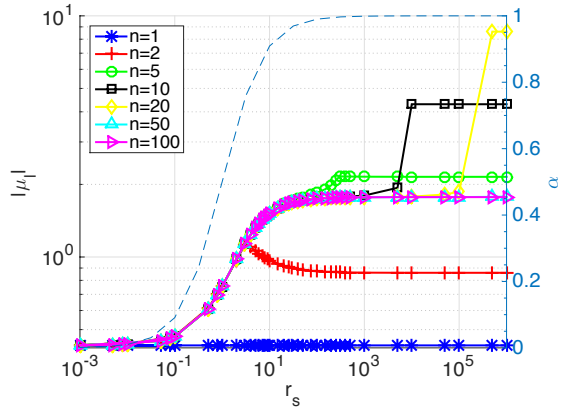

Figure 3.13: (m=4) This figure shows the variation of maximum stable value of $|\mu_\ell| = \Delta t_\ell \times |\lambda(\mathbf{A})|$ (see Eq. 3.20) as the *splitting ratio*, r_s (see Eq. 3.20), is varied for different values of timestep ratio (n), for conservative spring-mass system studied in Section 3.2.1. Variation of α (Eq. 3.21) is also shown. When the fraction of the forcing updated by smaller timestep, Δt_s, is smaller, *i.e* when $r_s \ll 1$, the stability is determined by the larger timestep, Δt_ℓ. The stability *criterion* that μ_s needs to satisfy is same as the stability *criterion* in the *synchronous* case ($n = 1$). As the fraction of the forcing updated by the smaller timestep (Δt_s), increases, μ_ℓ and μ_s undergo transition from larger timestep, Δt_ℓ, determining stability to the smaller timestep, Δt_s, determining stability.

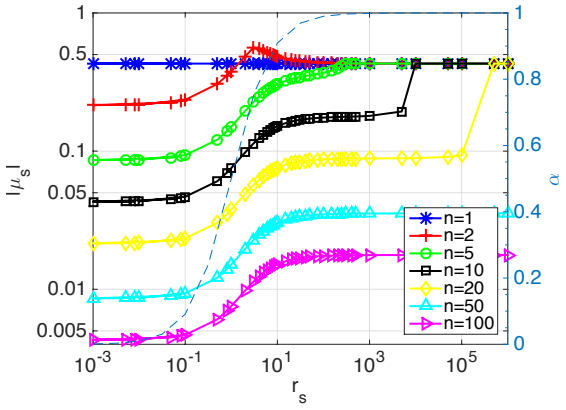

Figure 3.14: (m=4) This figure shows the variation of maximum stable value of $|\mu_\mathrm{s}| = \Delta t_\mathrm{s} \times |\lambda(\mathbf{A})|$ (see Eq. 3.20) as the *splitting ratio*, r_s (see Eq. 3.20), is varied for different values of timestep ratio (n), for conservative spring-mass system studied in Section 3.2.1. Variation of α (Eq. 3.21) is also shown. The smaller timestep, Δt_s, seems to determine the stability only when larger fraction of the forcing is updated using the smaller timestep, *i.e* when $r_s \gg 1$, especially after μ_s and μ_ℓ undergo transition. The stability *criterion* that μ_s needs to satisfy is same as it is for its *synchronous* counterpart (n=1).

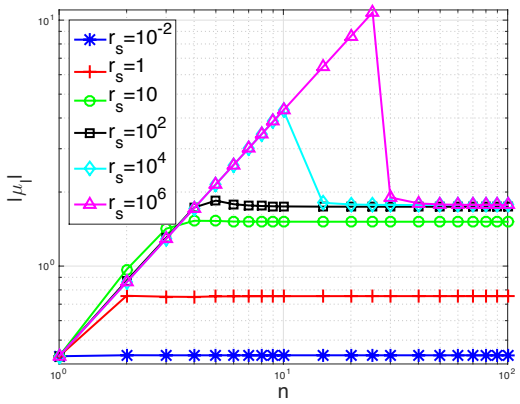

Figure 3.15: (m=4) This figure shows the variation of maximum stable value of $|\mu_\ell|$ as the *timestep ratio*, n ($= \Delta t_\ell / \Delta t_s$), is varied for different values of the *splitting ratio* r_s, in the spring-mass system studied in Section 3.2.1. When the fraction of the forcing that is updated by smaller timestep is large enough, the maximum stable value of larger timestep, Δt_ℓ, grows with n and peaks before transitioning to a value similar to $n = 1$. When $r_s \ll 1$, Δt_ℓ determines the stability.

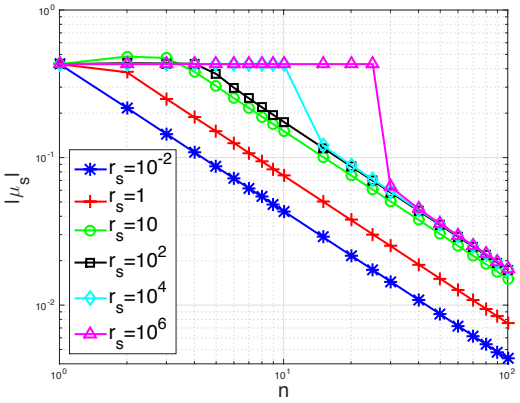

Figure 3.16: (m=4) This figure shows the variation of maximum stable value of $|\mu_s|$ as the *timestep ratio*, n ($= \Delta t_\ell / \Delta t_s$), is varied for different values of the *splitting ratio* r_s, in the spring-mass system studied in Section 3.2.1. When the fraction of the forcing that is updated by smaller timestep is large enough, smaller timestep (Δt_s) determines stability, before μ_s and μ_ℓ undergo *transition*.

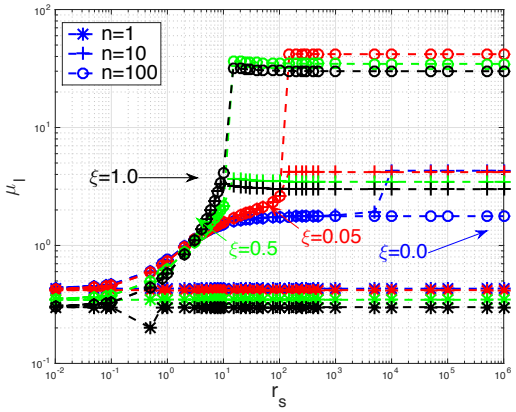

Figure 3.17: (m=4) This figure shows the variation of maximum stable value of $|\mu_\ell| = \Delta t_\ell \times |\lambda(\mathbf{A})|$ (see Eq. 3.20) as the *splitting ratio*, r_s (see Eq. 3.20), is varied for different values of *timestep ratio* (n) and for different values of *damping ratio*, $\xi = \{0.0, 0.05, 0.5, 1.0\}$, in spring-mass-damper systems studied in Section 3.2.1. For systems with damping, the timestep that determines stability is completely determined by the *splitting ratio*, r_s. Smaller fractions of forcing, that is updated using the smaller timestep, are needed with increasing damping in the system at the *transition* point.

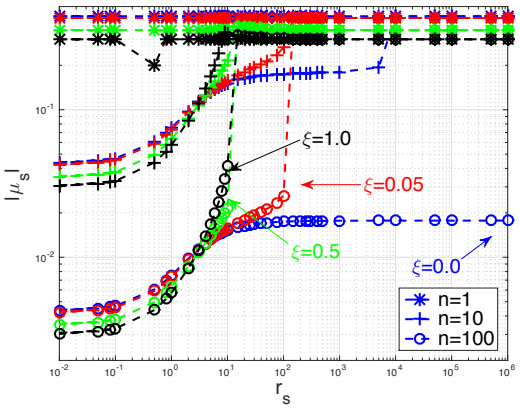

Figure 3.18: (m=4) This figure shows the variation of maximum stable value of $|\mu_\mathrm{s}| = \Delta t_\mathrm{s} \times |\lambda(\mathbf{A})|$ (see Eq. 3.20) as the *splitting ratio*, r_s (see Eq. 3.20), is varied for different values of *timestep ratio* (n) and for different values of *damping ratio*, $\xi = \{0.0, 0.05, 0.5, 1.0\}$, in spring-mass-damper systems studied in Section 3.2.1. For systems with damping, the timestep that determines stability is completely determined by the *splitting ratio*, r_s. Smaller fractions of forcing that is updated using the smaller timestep are needed with increasing damping in the system, for the smaller timestep to determine stability.

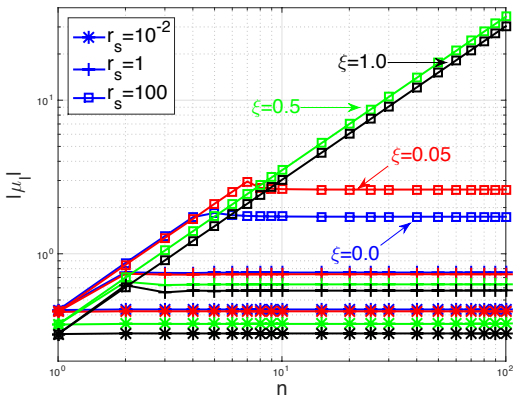

Figure 3.19: (m=4) This figure shows the variation of maximum stable value of $|\mu_\ell|$ as the *timestep ratio*, n ($= \Delta t_\ell / \Delta t_\mathrm{s}$), is varied for different values of the *splitting ratio* r_s and for different values of *damping ratio*, $\xi = \{0.0, 0.05, 0.5, 1.0\}$, in the spring-mass-damper system studied in Section 3.2.2. With resonable damping in the system and with the fraction of the forcing that is updated by smaller timestep being large enough, the maximum stable larger timestep, Δt_ℓ, grows monotonically as *timestep ratio* (n) increases. Indicating that with reasonable damping, smaller timestep, Δt_s, completely determines stability.

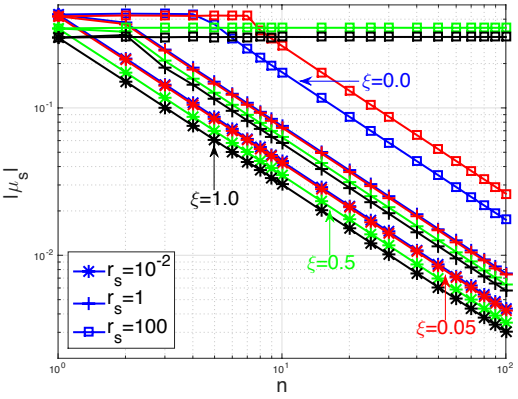

Figure 3.20: (m=4) This figure shows the variation of maximum stable value of $|\mu_\mathrm{s}|$ as the *timestep ratio*, n ($= \Delta t_\ell / \Delta t_\mathrm{s}$), is varied for different values of the *splitting ratio* r_s, in the spring-mass system studied in Section 3.2.1. In the case of systems with reasonable damping, the *splitting ratio* (r_s) completely determines which timestep governs stability.

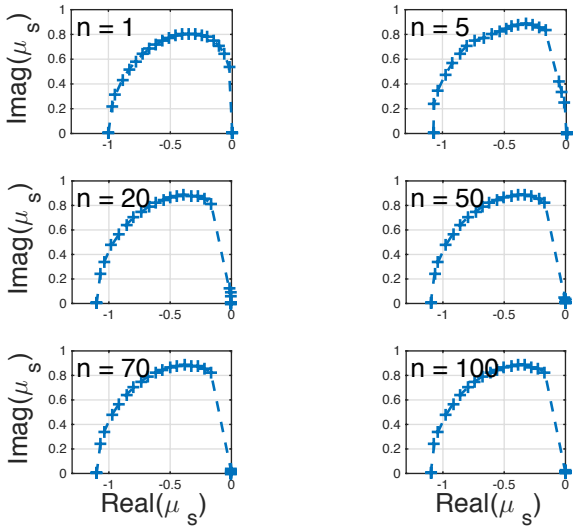

Figure 3.21: (m=2) Complex stability region of μ_s with different values of *timestep ratio*, n, when $r_s = 100$, are shown. The conservative eigen modes are the first to undergo *transition* and extend to slightly damped eigen modes with increasing the *timestep ratio* (n).

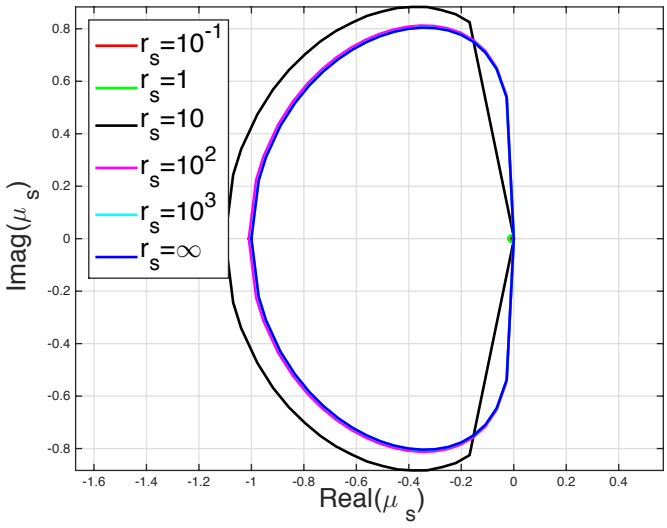

Figure 3.22: (m=2) Complex stability region of μ_s is shown for different values of the *splitting ratio*, r_s (see Eq. 3.20), and when $n = 100$. Larger the damping ratio in an eigen mode, smaller value of *splitting ratio*, r_s, is sufficient to allow the choice of larger timestep to be independent.

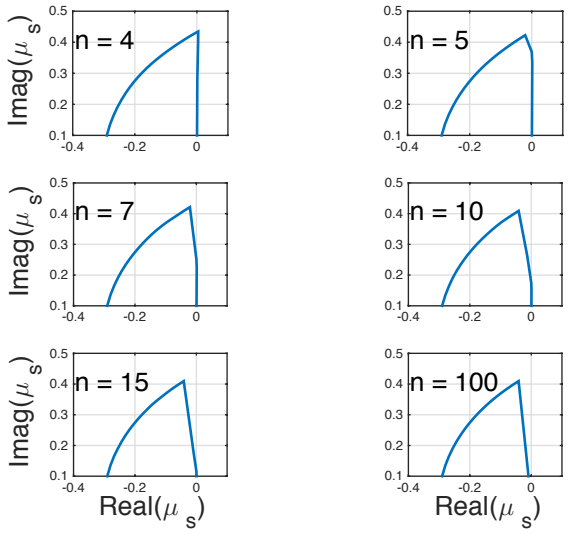

Figure 3.23: (m=4) Complex stability region of μ_s with different values of *timestep ratio*, n, when $r_s = 100$, are shown. The conservative eigen modes are the first to undergo *transition* and extend to slightly damped eigen modes with increasing the *timestep ratio* (n).

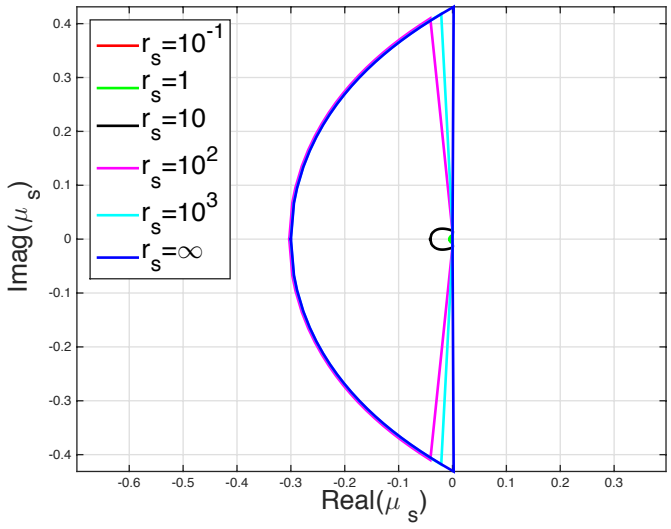

Figure 3.24: (m=4) Complex stability region of μ_s is shown for different values of the *splitting ratio*, r_s (see Eq. 3.20), and when $n = 100$. Larger the damping ratio in an eigen mode, smaller value of *splitting ratio*, r_s, is sufficient to allow the choice of larger timestep to be independent.

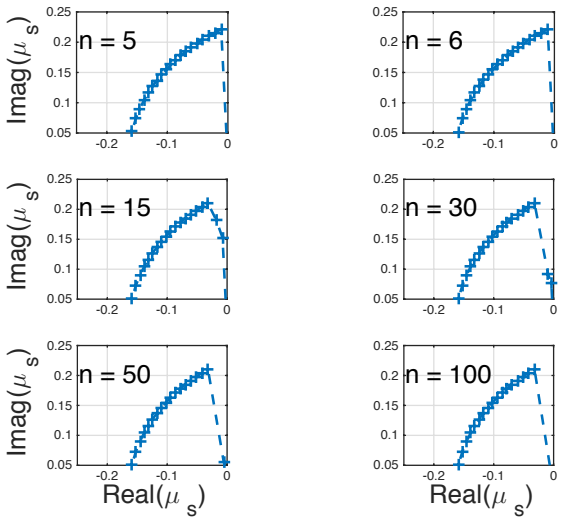

Figure 3.25: (m=5) Complex stability region of μ_s with different values of *timestep ratio*, n, when $r_s = 100$, are shown. The conservative eigen modes are the first to undergo *transition* and extend to slightly damped eigen modes with increasing the *timestep ratio* (n).

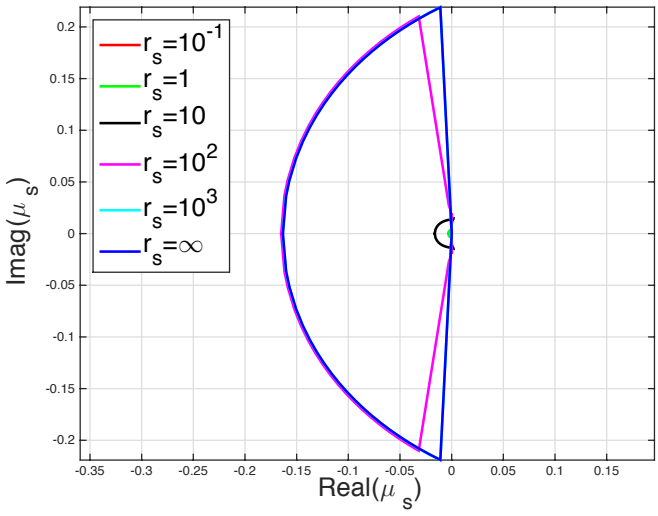

Figure 3.26: (m=5) Complex stability region of μ_s is shown for different values of the *splitting ratio*, r_s (see Eq. 3.20), and when $n = 100$. Larger the damping ratio in an eigen mode, smaller value of *splitting ratio*, r_s, is sufficient to allow the choice of larger timestep to be independent.

Chapter 4

Time Step Selection

In this chapter, we develop a time-step selection (TSS) method for solving multiple-time-scale ODEs using MASM. There are many such variable timestep implementations that have been studied [43, 40, 42, 56, 64, 59, 8, 57]. Here we only explore a basic implementation of TSS using MASM. We also present a sparse implementation of MASM useful for large scale simulations as well as parallel implementation.

The outline of this chapter is as follows: Section 4.1 presents the sparse implementation of MASM. In section 4.2 we develop the TSS scheme for MASM. We numerically study the performance of this TSS method on two examples: (1) a linear spring-mass system, (2) sparsely-coupled (i.e., has fewer non-diagonal terms in the Jacobian, $\frac{d\mathbf{F}}{d\mathbf{X}}$) nonlinear aerosol problem in Section 4.3. We will see that MASM, while being as efficient as a synchronous method when modeling single timescale systems, improves the efficiency by several orders when applied to problems with varied time scales.

4.1 Sparse Implementation of MASM

In the implementation of MASM (see Algorithm 1), at each timestep, the entire state vector, $\mathbf{X}(t)$, is updated, even if only a small section of the solution is used for forcing component evaluation. This results in several parts of the solution being updated several times before being employed in a force field evaluation. Sparse implementation of MASM avoids this inefficiency. Parts of the solution that affects the component force field to be evaluated, are only updated. Such an implementation, besides avoiding unnecessary updates of other

parts of the solution, also leads to easier implementation. It also avoids computational overheads related to updating entire state vector at each step, which can be large for large scale simulations. In addition, you will also see that sparse implementation of MASM is suitable for parallel computing.

In this implementation, instead of updating full state vectors we updated component state vectors given by,

$$\langle \mathbf{\Phi}^{[j]} \rangle_p = \begin{cases} \langle \mathbf{\Phi} \rangle_p & \text{when } p \in \mathfrak{S}^{[j]} \\ 0 & \text{when } p \notin \mathfrak{S}^{[j]} \end{cases} \tag{4.1}$$

where $\mathfrak{S}^{[j]}$ is a set of vector element indices of non-zero quantities in $\mathbf{F}^{[j]}$. Essentially, a component state vector, say $\mathbf{\Phi}_p^{[j]}$, is the part of the state vector whose dynamics is affected by a component force vector, *i.e* $\mathbf{F}_p^{[j]}$ in the case of $\mathbf{\Phi}_p^{[j]}$. The corresponding component vector field evaluations are $\mathbf{F}_p^{[j]} = \mathbf{F}^{[j]}(\mathbf{\Phi}_p, t_p^{[j]}) = \mathbf{F}^{[j]}(\{\mathbf{\Phi}_p^{[q]}\}_{q \in \mathcal{K}^{[j]}}, t_p^{[j]})$, where,

$$\mathcal{K}^{[j]} = \left\{ q \,\middle|\, \frac{d\mathbf{F}^{[j]}}{d\mathbf{\Phi}^{[q]}} \neq \mathbf{0} \right\}, \tag{4.2}$$

i.e $\mathcal{K}^{[j]}$ is a set of component indices of component state vectors that effect $\mathbf{F}^{[j]}$. In sparse systems, $\mathbf{\Phi}^{[j]}$ and $\mathbf{F}^{[j]}$ have several quantities which are identically zero, therefore we define local vectors as,

$$\boldsymbol{\phi}^{[j]} = \mathcal{A}^{[j]} \mathbf{\Phi} \tag{4.3}$$

$$\mathbf{f}^{[j]} = \mathcal{A}^{[j]} \mathbf{F}^{[j]} = \mathcal{A}^{[j]} \mathbf{F} \tag{4.4}$$

where, $\boldsymbol{\phi}^{[j]}$ is the j^{th} component local state vector, $\mathbf{f}^{[j]}$ is j^{th} component local force vector and

$$[\mathcal{A}^{[j]}]_{p,q} = \begin{cases} 1 & q \in \mathfrak{S}^{[j]} \& q \text{ is } p^{th} \text{non-zero quantity of } \mathbf{F}^{[j]} \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

$\mathcal{A}^{[j]}$ is a vector operator which returns a vector of the elements of the argument vector, where the corresponding elements in $\mathbf{F}^{[j]}$ have a non-zero value. For simplicity we assume that, $\mathfrak{S}^{[j]}$, $\mathcal{K}^{[j]}$ and $\mathcal{A}^{[j]}$ do not change with time.

Note that $\bigcup_j \mathfrak{S}^{[j]} = \{1, \ldots, N\}$ where N is the dimensionality of $\mathbf{\Phi}$ and dimensionality of $\boldsymbol{\phi}^{[j]}$ and $\mathbf{f}^{[j]}$ are much smaller than N for sparse systems. Note also that when $\bigcap_j \mathfrak{S}^{[j]} = \{1, \ldots, N\}$, then sparse implementation of MASM is equivalent to the earlier implementation of MASM introduced in Section 2.1.3, which is suitable for modeling vector field splitting problems. In the case when $\bigcap_j \mathfrak{S}^{[j]} = \{\}$ (null set), its equivalent to variable splitting problem. Therefore, the sparse implementation of MASM generalizes both vector field splitting and variable splitting problems and exploits the degree of sparsity in the problem in eliminating unnecessary computations and also reducing the computational over head in dealing with large vectors.

An m-step MASM integrator proceeds in a sequence of steps, indexed by k, by updating component local state vectors $\boldsymbol{\phi}^{[j]}$ by,

$$\boldsymbol{\phi}^{[j]}_{\ell^{[j]}(k+1)} = \boldsymbol{\phi}^{[j]}_{\ell^{[j]}(k)} + \Delta t^{[j]}_{\ell^{[j]}(k)} \sum_{i=0}^{m-1} \langle \boldsymbol{\beta}^{[j]}_k \rangle_i \mathbf{f}^{[j]}_{\ell^{[j]}(k)-i} \tag{4.6}$$

where $\Delta t^{[j]}_{\ell^{[j]}(k)}$ are component timesteps and $\boldsymbol{\beta}^{[j]}_k$ can be determined by Eq. 2.19. At step k we assume that for each component j we have m previous local component states $\boldsymbol{\phi}^{[j]}_{\ell^{[j]}(k)-i}$, $i = 0, \ldots, m-1$, that represent the component state at times $t^{[j]}_{\ell^{[j]}(k)-i}$, $i = 0, \ldots, m-1$. The corresponding component local vector field evaluations are $\mathbf{f}^{[j]}_{\ell^{[j]}(k)-i} = \mathcal{A}^{[j]} \mathbf{F}^{[j]}_{\ell^{[j]}(k)-i}$.

Given *component timesteps* $\Delta t^{[j]}_{\ell^{[j]}(k)}$ for each component j, we choose the component j_k such that,

$$t^{[j_k]}_{\ell^{[j]}(k)} + \Delta t^{[j_k]}_{\ell^{[j]}(k)} = \min_{j \in \{j | t^{[j]}_{\ell^{[j]}(k)} < t_f\}} \left(t^{[j]}_{\ell^{[j]}(k)} + \Delta t^{[j]}_{\ell^{[j]}(k)} \right). \tag{4.7}$$

Now the component local state vector $\boldsymbol{\phi}^{[j_k]}$ is updated using,

$$\boldsymbol{\phi}^{[j_k]}_{\ell^{[j_k]}(k+1)} = \boldsymbol{\phi}^{[j_k]}_{\ell^{[j_k]}(k)} + \Delta t^{[j_k]}_{\ell^{[j_k]}(k)} \sum_{i=0}^{m-1} \beta^{[j_k]}_{\ell^{[j_k]}(k)-i} \mathbf{f}^{[j_k]}_{\ell^{[j_k]}(k)-i}. \tag{4.8}$$

Since the j_k^{th} component force field needs to be evaluated, $\{\boldsymbol{\phi}^{[j]}\}_{j\in\mathcal{K}^{[j_k]}-j_k}$ are temporarily updated by,

$$\boldsymbol{\phi}^{[j]}_{\ell^{[j]}(k+1)} = \boldsymbol{\phi}^{[j]}_{\ell^{[j]}(k)} + \left(t^{[j_k]}_{\ell^{[j_k]}(k)} + \Delta t^{[j_k]}_{\ell^{[j_k]}(k)} - t^{[j]}_{\ell^{[j]}(k)}\right) \sum_{i=0}^{m-1} \langle \boldsymbol{\beta}^{[j]}_k \rangle_i \mathbf{f}^{[j]}_{\ell^{[j]}(k)-i}. \tag{4.9}$$

These temporary values are discarded after $\mathbf{f}^{[j_k]}_{\ell^{[j_k]}(k+1)} = \mathcal{A}^{[j_k]}\mathbf{F}^{[j_k]}(\{\boldsymbol{\phi}^{[j]}_{\ell^{[j]}(k+1)}\}_{j\in\mathcal{K}^{[j_k]}}, t^{[j_k]}_{\ell^{[j_k]}(k+1)})$ is evaluated. As a result, the computational costs are minimized exploiting the sparsity in the system, while generating numerical solution of the same accuracy as in Section 2.1.3.

Similar to earlier implementation of MASM, we need a *starting procedure* to compute the first $m-1$ steps for each component j. We also use a standard interpolation to evaluate the solution at particular times, including the final time. The coefficients $\langle \boldsymbol{\beta}^{[j]}_k \rangle_i$ must generally be determined at each step, as described in Section 2.2, as the local incremental step sizes are varying. The data that must be maintained for sparse implementation of MASM is the current component local state vector, $\boldsymbol{\phi}^{[j]}_{\ell^{[j]}(k)}$, and the m previous function evaluations, $\mathbf{f}^{[j]}_{\ell^{[j]}(k)-i}$, $i = 0, \ldots, m-1$, for each system component $j = 1, \ldots, N$.

The algorithmic description of sparse implementation of MASM is given in Algorithm (2) and a graphical depiction of the variable indexing is shown in Figure 2.1 (right panel). Note that Algorithm (2) is written without in-place updating of variables (except k). In practical implementations, only single $m \times \dim \mathfrak{S}_j$ arrays are stored for times $t^{[j]}_{k,i}$, states $\boldsymbol{\phi}^{[j]}_{k,i}$, and components $\mathbf{f}^{[j]}_{k,i}$ and these are updated in-place.

Algorithm 2 MASM sparse algorithm (m-step explicit Adamstype)

Input: final time t_{f}, initial component times $t^{[j]}_{\ell^{[j]}(0)-i}$ and states $\boldsymbol{\phi}^{[j]}_{\ell^{[j]}(0)-i} (= \mathcal{A}^{[j]}\boldsymbol{\Phi}^{[j]}_{\ell^{[j]}(0)}$, refer Eq. (4.3)) for $i = 0, \ldots, m-1$ and $j = 1, \ldots, N_{\mathrm{f}}$
Initialize: $k \leftarrow 0$, $t_k \leftarrow \max_j t^{[j]}_{\ell^{[j]}(0)}$, with $\boldsymbol{\Phi}_k (= \sum_j \boldsymbol{\Phi}^{[j]}_{\ell^{[j]}(0)})$ the corresponding state
Initialize: choose per-component timesteps $\Delta t^{[j]}_{\ell^{[j]}(0)}$
Initialize: $\mathbf{f}^{[j]}_{\ell^{[j]}(0)-i} \leftarrow \mathcal{A}^{[j]}\mathbf{F}^{[j]}(\{\boldsymbol{\phi}^{[p]}_{\ell^{[p]}(0)-i}\}_{p\in\mathcal{K}^{[j]}}, t^{[j]}_{\ell^{[j]}(0)-i})$ for $i = 0, \ldots, m-1$ and $j = 1, \ldots, N_{\mathrm{f}}$
while $t_k < t_{\mathrm{f}}$ **do**
  $t_{k+1} \leftarrow \min_p \left(t^{[p]}_{\ell^{[p]}(k)} + \Delta t^{[p]}_{\ell^{[p]}(k)}\right) \forall\ p$ such that $\mathrm{t}^{[p]}_{\ell^{[p]}(k)} < \mathrm{t}_f$
  $j_k \leftarrow p$ such that $t_{k+1} = t^{[p]}_{\ell^{[p]}(k)} + \Delta t^{[p]}_{\ell^{[p]}(k)}$
  while $p \in \mathcal{K}^{[j_k]}$(refer Eq. (4.2)) **do**
    $\tau^{[p]}_{\ell^{[p]}(k)-i} \leftarrow (t^{[p]}_{\ell^{[p]}(k)-i} - t^{[j_k]}_{\ell^{[j_k]}(k)})/(t^{[j_k]}_{\ell^{[j_k]}(k)} + \Delta t^{[j_k]}_{\ell^{[j_k]}(k)} - t^{[p]}_{\ell^{[p]}(k)})$ for $i = 0, \ldots, m-1$
    $(\mathbf{V}^{[p]}_k)_{p,i} \leftarrow (\tau^{[p]}_{\ell^{[p]}(k)-i})^p$ for $i = 0, \ldots, m-1$
    $\langle \mathbf{R} \rangle_i = \frac{1}{i+1}$ for $i = 0, \ldots, m-1$
    Solve $\mathbf{V}^{[p]}_k \boldsymbol{\beta}^{[p]}_k = \mathbf{R}$ for the vector $\boldsymbol{\beta}^{[p]}_k$
    $\tilde{\boldsymbol{\phi}}^{[p]}_{\ell^{[p]}(k+1)} \leftarrow \boldsymbol{\phi}^{[p]}_{\ell^{[p]}(k)} + (t^{[j_k]}_{\ell^{[j_k]}(k)} + \Delta t^{[j_k]}_{\ell^{[j_k]}(k)} - t^{[p]}_{\ell^{[p]}(k)}) \sum_{i=0}^{m-1} \langle \boldsymbol{\beta}^{[p]}_k \rangle_i \mathbf{f}^{[p]}_{\ell^{[p]}(k)-i}$
  end while
  $t^{[j_k]}_{\ell^{[j_k]}(k+1)} \leftarrow t_{k+1}$, $\boldsymbol{\phi}^{[j_k]}_{\ell^{[j_k]}(k+1)} \leftarrow \tilde{\boldsymbol{\phi}}^{[j_k]}_{\ell^{[j_k]}(k+1)}$, $\mathbf{f}^{[j_k]}_{\ell^{[j_k]}(k+1)} \leftarrow \mathcal{A}^{[j_k]}\mathbf{F}^{[j_k]}(\{\tilde{\boldsymbol{\phi}}^{[p]}_{\ell^{[p]}(k+1)}\}_{p\in\mathcal{K}^{[j_k]}}, t_{k+1})$
  $t^{[j_k]}_{\ell^{[j_k]}(k+1)-i} \leftarrow t^{[j_k]}_{\ell^{[j_k]}(k)-(i-1)}$, $\boldsymbol{\phi}^{[j_k]}_{\ell^{[j_k]}(k+1)-i} \leftarrow \boldsymbol{\phi}^{[j_k]}_{\ell^{[j_k]}(k)-(i-1)}$, $\mathbf{f}^{[j_k]}_{\ell^{[j_k]}(k+1)-i} \leftarrow \mathbf{f}^{[j_k]}_{\ell^{[j_k]}(k)-(i-1)}$ for $i = 1, \ldots, m-1$
  update timestep $\Delta t^{[j_k]}_{\ell^{[j_k]}(k+1)}$
  $k \leftarrow k+1$
end while
Finalize: compute final state $\boldsymbol{\Phi}_{\mathrm{f}}$ at time t_{f} with mth order interpolant

4.2 Timestep Size Selection

The theory of solving ODEs numerically is well developed and TSS play a very important role in realizing the optimum choice of the timestep in real time for greater efficiency. An optimum timestep choice changes with the ODE and the physics it represents and for non-autonomous systems, the optimum timestep changes with time. Rarely does it happen that the user would know in advance the optimum timestep and its time dependency. TSS thereby not only improves the computational efficiency, but also reduces human time to obtain the numerical solution efficiently. Many of the present day ODE solvers use strategies and

ideas developed by Krogh[41] and Gear[19]. For more detailed discussion, one can refer [58] and [22].

In this section, we will develop the TSS suitable for MASM. The theory behind TSS with changing dynamics is reasonably well understood. However, in the case of MASM, we are looking to determine more than one timestep. Each component timestep choice now not only depends upon the overall dynamics, but also the dynamics of the component force vector fields and/or their corresponding component timestep values. Note that the dependence of optimum component timestep on the dynamics of the respective component force field is similar to that in *synchronous* methods. Essentially, the optimum timestep in *synchronous* methods is chosen such that certain accuracy is maintained in the numerical solution. It is a proven fact that, as the upper bound to *local discretization error* $(\tau) \to 0$, the global error in the numerical solution is $\sim \mathcal{O}(\tau)$ [58]. As a result, the timestep selection is made based on the *local discretization error*, which ensures a certain order of accuracy in the numerical solution. Therefore understanding the interdependence of optimal component timesteps is critical for developing an effective TSS strategy for modeling ODEs using MASM. Following *lemma* will help us in doing the same.

Lemma 3. Local discretization error *when* Adams *type m-step MASM applied to Eq. 2.4 is given by,*

$$\mathbf{L}_m^{\Delta t_k}(t_k) = \sum_j \mathbf{L}_m^{[j]\Delta t_k}(t_k) + \mathcal{O}\left(\Delta t_k\right)^{m+2} \tag{4.10}$$

where,

$$\mathbf{L}_m^{[j]\Delta t_k}(t_k) = C_k^{[j]} \mathbf{F}^{[j](m)}(t_k) \Delta t_k^{m+1} + \mathcal{O}\left(\Delta t_k\right)^{m+2}, \tag{4.11}$$

$$C_k^{[j]} = \frac{1}{(m+1)!}\left[1-(m+1)\left(\sum_{i=0}^{m-1}\langle\boldsymbol{\beta}_k^{[j]}\rangle_i\left(\tau_{\ell^{[j]}(k)-i}^{[j]}\right)^m\right)\right], \tag{4.12}$$

and

$$\tau_{\ell^{[j]}(k)-i}^{[j]} = \left(\frac{t_{\ell^{[j]}(k)-i}^{[j]} - t_k}{\Delta t_k}\right). \tag{4.13}$$

Proof. The Linear difference operator, in case of MASM is given by,

$$\mathbf{L}_m^{\Delta t_k}(t_k) = \mathbf{\Gamma}^{\Delta t_k}(\mathbf{\Phi}_k) - \mathbf{\Psi}_m^{\Delta t_k}(\mathbf{\Phi}_k), \tag{4.14}$$

where $\mathbf{\Gamma}^{\Delta t_k} : \mathbb{R}^N \to \mathbb{R}^N$ is a one-step integrator for the exact solution and $\mathbf{\Psi}_m^{\Delta t_k}$ is the one-step integrator for numerical solution of *Adams* type *m-step* MASM.

$$\begin{aligned}
\mathbf{L}_m^{\Delta t_k}(t_k) &= \left(\mathbf{\Psi}_{m+1}^{\Delta t_k}(\mathbf{\Phi}_k) - \mathbf{\Psi}_m^{\Delta t_k}(\mathbf{\Phi}_k)\right) + \left(\mathbf{\Gamma}^{\Delta t_k}(\mathbf{\Phi}_k) - \mathbf{\Psi}_{m+1}^{\Delta t_k}(\mathbf{\Phi}_k)\right) &(4.15)\\
&= \left(\mathbf{\Psi}_{m+1}^{\Delta t_k}(\mathbf{\Phi}_k) - \mathbf{\Psi}_m^{\Delta t_k}(\mathbf{\Phi}_k)\right) + \mathcal{O}\left(\Delta t_k\right)^{m+2}\\
&= \left(\mathbf{\Phi}_k + \Delta t_k \sum_j \sum_{i=0}^{m}\langle\boldsymbol{\beta'}_k^{[j]}\rangle_i \sum_{p=0}^{\infty}\mathbf{F}^{[j](p)}(t_k)\left(\tau_{\ell^{[j]}(k)-i}^{[j]}\right)^p \frac{\Delta t_k^p}{p!}\right)\\
&\quad - \left(\mathbf{\Phi}_k + \Delta t_k \sum_j \sum_{i=0}^{m-1}\langle\boldsymbol{\beta}_k^{[j]}\rangle_i \sum_{p=0}^{\infty}\mathbf{F}^{[j](p)}(t_k)\left(\tau_{\ell^{[j]}(k)-i}^{[j]}\right)^p \frac{\Delta t_k^p}{p!}\right) + \mathcal{O}\left(\Delta t_k\right)^{m+2} &(4.16)
\end{aligned}$$

where, $\tau_{\ell^{[j]}(k)-i}^{[j]}$ is given by Eq. 4.13, $\boldsymbol{\beta'}_k^{[j]}$ satisfy $\sum_{i=0}^{m}\langle\boldsymbol{\beta'}_k^{[j]}\rangle_i\left(\tau_{\ell^{[j]}(k)-i}^{[j]}\right)^p = \frac{1}{p+1}$ for $p = 0,\ldots,m$ and similarly, $\langle\boldsymbol{\beta}_k^{[j]}\rangle_i$ satisfy $\sum_{i=0}^{m-1}\langle\boldsymbol{\beta}_k^{[j]}\rangle_i\left(\tau_{\ell^{[j]}(k)-i}^{[j]}\right)^p = \frac{1}{p+1}$ for $p = 0,\ldots,m-1$. Now using the definitions of $\boldsymbol{\beta'}_k^{[j]}$ and $\boldsymbol{\beta}_k^{[j]}$, the linear difference operator (4.16) can be reduced to,

$$\mathbf{L}_m^{\Delta t_k}(t_k) = \Delta t_k \left(\sum_j \mathbf{F}^{[j](m)}(t_k) \frac{(\Delta t_k)^m}{m+1!} + \mathcal{O}\left(\Delta t_k\right)^{m+1} \right)$$
$$- \Delta t_k \left(\sum_j \left(\sum_{i=0}^{m-1} \langle \boldsymbol{\beta}_k^{[j]} \rangle_i \left(\tau_{\ell^{[j]}(k)-i}^{[j]} \right)^m \right) \mathbf{F}^{[j](m)}(t_k) \frac{\Delta t_k^m}{m!} + \mathcal{O}\left(\Delta t_k\right)^{m+1} \right) + \mathcal{O}\left(\Delta t_k\right)^{m+2}, \tag{4.17}$$

$$= \sum_j \left(C_k^{[j]} \mathbf{F}^{[j](m)}(t_k) \frac{(\Delta t_k)^{m+1}}{m+1!} \right) + \mathcal{O}\left(\Delta t_k\right)^{m+2}, \tag{4.18}$$

where,

$$C_k^{[j]} = \frac{1}{(m+1)!} \left[1 - (m+1) \left(\sum_{i=0}^{m-1} \langle \boldsymbol{\beta}_k^{[j]} \rangle_i \left(\tau_{\ell^{[j]}(k)-i}^{[j]} \right)^m \right) \right]. \tag{4.19}$$

Hence proves Eqs. (4.10-4.12).

□

The expression in Eq. 4.10 highlights only the leading order term in the linear difference operator, which is a good approximation. It also assumes that the component timesteps, $\Delta t_k^{[j]}$, are small enough such that component force fields, $\mathbf{F}^{[j]}$, do not vary much, which is a standard assumption made in numerical analysis. It also helps decouple the contributions from each component force fields.

Lets recall the expression for *local discretization error*, which is a standard textbook result in the case of ABM (See [28]), which is given by,

$$\mathbf{L}_m^{\Delta t_k}(t_k) = C_{m+1} \mathbf{F}^{(m)}(\eta_k) \left(\Delta t_k\right)^{m+1}, \tag{4.20}$$

where $C_{m+1} = \frac{1}{(m+1)!} \left(1 - (m+1) \sum_{i=0}^{m-1} \beta_i (-i)^m\right)$ and $\eta_k \in [t_k, t_{k+1}]$. Note that β_i satisfy $\sum_{i=0}^{m-1} \beta_i (-i)^n = 1/(n+1)$, where $n = 0, \ldots, m-1$.

To understand the efficacy of ABM in modeling multiscale problems, we consider linear

systems (2.4), which are assumed to be *autonomous*, *i.e* $\mathbf{F}(t) = \mathbf{A}\mathbf{X}(t)$. We also assume that $\mathbf{A}$ is diagonalizable where $(\lambda_i, \boldsymbol{\nu}_i)$ are the eigen pairs. Then one can express

$$\mathbf{F}^{(m)}(t_k) = \sum_i c_i(t_k)\lambda_i^{m+1}\boldsymbol{\nu}_i, \tag{4.21}$$

where $\boldsymbol{\Phi}_k = \sum_i c_i(t_k)\boldsymbol{\nu}_i$ and $\mathbf{F}^{(m)} = d^m\mathbf{F}/dt^m$. Using which the linear difference operator (Eq. 4.20) can be expressed as,

$$\mathbf{L}_m^{\Delta t_k}(t_k) = C_{m+1}\left(\sum_i c_i(t_k)\lambda_i^{m+1}\boldsymbol{\nu}_i\right)\Delta t_k^{m+1} + \mathcal{O}\left(\Delta t_k\right)^{m+2}. \tag{4.22}$$

Linear difference operator in Eq. 4.22 shows the contribution of each eigen component to *local discretization error*. In case of *stiff* systems where $\max\{\lambda_i\} \gg \min\{\lambda_i\}$, the contribution from larger eigen values dominate and hence determine the timestep, while the contribution from smaller eigenvalues is small or negligible. This has been the drawback of traditional single timestep methods.

In the case of *Adams m-step* MASM method, the linear difference operator in Eq. 4.10 has a contribution from each component force vector field. The contribution from each component force field, $\mathbf{L}_m^{[*]\Delta t_k}(t_k)$, for the time interval $[t_k, t_{k+1}]$ is a scaled version of the component linear difference operator, given by,

$$\mathbf{L}_m^{[j]}(t^{[j]}_{\ell^{[j]}(k)}) = C^{[j]}_{\ell^{[j]}(k)}(\Delta t^{[j]}_{\ell^{[j]}(k)})\mathbf{F}^{[j](m)}_{\ell^{[j]}(k)}\left(\Delta t^{[j]}_{\ell^{[j]}(k)}\right)^{m+1} + \mathcal{O}\left(\Delta t^{[j]}_{\ell^{[j]}(k)}\right)^{m+2} \tag{4.23}$$

where,

$$C^{[j]}_{\ell^{[j]}(k)}(\Delta t^{[j]}_{\ell^{[j]}(k)}) = \frac{1}{(m+1)!}\left[1 - (m+1)\left(\sum_{i=0}^{m-1}\langle\boldsymbol{\beta}_k^{[j]}\rangle_i\,(-i)^m\right)\right]. \tag{4.24}$$

Note that by replacing $\Delta t^{[j]}_{\ell^{[j]}(k)}$ and $\mathbf{F}^{[j](m)}_{\ell^{[j]}(k)}$ with Δt_k and $\mathbf{F}_k^{(m)}$ respectively, Eqs. (4.23,4.24) can be reduced to Eqs. (4.20). Since the timestep Δt_k is smaller than the component timesteps, $\Delta t_k^{[j]}$, the contribution from each component force fields are accordingly scaled.

Note that, many a times the contribution from component force fields are updated several times within the component timestep, which can also be see in Fig. 2.1 (right figure). The same issue has also been touched upon in Section 4.1. When all the component timesteps are same or when all component timesteps are equivalent to timestep, Δt_k, one can see that Eq. 4.10 or Eq. 4.11 respectively reduces to Eq. 4.20.

Now in the case of the linear, autonomous and diagonalizable systems (2.4) with split force vector field, Eq. 4.10 can be expressed as,

$$\mathbf{L}_m^{\Delta t_k}(t_k) = \sum_j C_k^{[j]} \left(\sum_{\{i|\boldsymbol{\nu}_i^T \mathbf{B}^{[j]} \boldsymbol{\nu}_i \neq 0\}} c_i^{[j]}(t_k) \lambda_i^{m+1} \boldsymbol{\nu}_i \right) (\Delta t_k)^{m+1} + \mathcal{O}\left(\Delta t_k\right)^{m+2} \tag{4.25}$$

where $C_k^{[j]}$ is defined by Eq. 4.12, and

$$\mathbf{F}^{[j](m)}(t) = \mathbf{B}^{[j]}\mathbf{X}(t) = \sum_{\{i|\boldsymbol{\nu}_i^T \mathbf{B}^{[j]} \boldsymbol{\nu}_i \neq 0\}} c_i^{[j]}(t) \lambda_i^{m+1} \boldsymbol{\nu}_i. \tag{4.26}$$

Since MASM uses more than one timestep, one can choose these component timesteps such that the contribution from different component force fields is of the same order. In case if the smaller eigen values of the system are a dominate contributors of local discretization error from a component force field, it allows the use of larger timestep for respective component force field resulting in computational savings while marginally changing the numerical accuracy of the solution.

MASM is a method which allows you to use arbitrarily large number of timesteps for modeling the ODEs (2.4), whose force vector field can be split. From Eq. 4.10, it is evident that by proper choice of component force fields and component timestep such that, $\mathbf{L}^{[j]}$ contribute equally to *local discretization error*, the inefficiency of *synchronous* methods while modeling stiff systems can be avoided.

TSS strategy developed here is based on component *local discretization error*, $\mathbf{L}_m^{[j]}$, where

in the component timesteps are chosen such that *local discretization error* is equally contributed by all components, *local discretization error*, and that the *local discretization error* does not exceed the specified error tolerance (τ). This results in the global error being $\mathcal{O}(\tau)$ as $\tau \to 0$ [58]. It would seem from Eq. 4.10 that, when optimum values of component timesteps are chosen, the contributions from each component force field to *local discretization error* will be the same. Since the contributions from each component force field at each time t_k is scaled by $(\Delta t_k/\Delta t_k^{[j]})$, the above approach at each timestep could lead to unnecessary frequent timestep changes of the component timesteps. We will see ahead that it also effects the accuracy of error estimation which assumes constant timestep. Therefore the component timesteps are chosen such that $\mathbf{L}_m^{\Delta t^{[j]}_{\ell^{[j]}(k)}}(t^{[j]}_{\ell^{[j]}(k)})$ are scaled. Depending upon the vector norm that is used, the component timesteps are chosen such that,

$$\left\| \mathbf{L}_m^{\Delta t^{[j]}_{\ell^{[j]}(k)}}(t^{[j]}_{\ell^{[j]}(k)}) \right\|_p \leq \begin{cases} \tau & p = 0 \\ \tau/N_{\mathrm{f}} & p \neq 0 \end{cases} \tag{4.27}$$

Similar to High-order multistep, TSS based MASM needs starting values and hence are not self starters. Often one needs to generate the starting values using other methods. By varying order and step size, one can also make MASM a self starter, but at the same time maintaining the required accuracy.

4.2.1 Convergence and Stability

In Section 2.3, we have proven the convergence (*Theorem* 2, *Theorem* 3) of numerical solution, obtained using MASM on variable grid where the component timesteps satisfy Eq. (2.30), as $h_{max} \to 0$ (see Eq. 2.30). However, when timestep choice is made such that *local discretization error* is below a specified error tolerance (τ), more relevant convergence of the numerical solution is when $\tau \to 0$. For any timestep adaptation, the above numerical convergence is achieved when as $\tau \to 0$, $h_{max} \to 0$ while satisfying Eq. (2.30). Its trivial to see that $h_{max} \to 0$ as $\tau \to 0$, however, the only scenario when Eq. (2.30) can still be violated

is when one or more component timesteps are driven to zero much faster than h_{max}. Which is unlikely given that time step selection strategy is such that contributions from component force fields to *local discretization error* is equally distributed.

As was discussed in Section 3.1, instability in a numerical solution occurs as timestep is driven to zero (see *Dahlquist I criterion*) or when the timestep is too large to make a reasonable approximation (see *Dahlquist II criterion*). In timestep adaptation, since the timestep choice is based on the local error, the instability due to larger timestep is avoided as whenever the error grows, the timestep accordingly is reduced. In cases where monotonous increase in error happen, the timestep are driven to zero and the simulations essentially halts. Also the instability due to larger timestep happens during a simulation. Instability due to smaller timestep, is seen when smaller timestep choice is made as $\tau \to 0$. The same has already been established in the numerical solution obtained using MASM (see *Theorem* 4). Since numerical convergence of MASM with time step selection strategy, is already established. That also proves the *Zero-Stability* of the numerical solution using MASM with time step selection strategy.

Both these results can also be extended to variable order methods, provided the maximal order is bounded.

4.2.2 Error Estimate

Estimation of *local discretization error* contributed by different component force fields is key for the timestep size selection strategy to be effective. From Lemma 3, its clear that the *local discretization error* from component force field can be estimated by,

$$
\begin{aligned}
\epsilon^{[j_k]}(\Delta t^{[j_k]}_{\ell^{[j_k]}(k)}) &= \left\| \mathbf{L}^{[j_k]}(\mathbf{\Phi}^{[j_k]}_{\ell^{[j_k]}(k)}, t^{[j_k]}_{\ell^{[j_k]}(k)}) \right\| \\
&= \| \mathbf{\Gamma}^{\Delta t^{[j_k]}_{\ell^{[j_k]}(k)}}(\mathbf{\Phi}^{[j_k]}_{\ell^{[j_k]}(k)}) - \mathbf{\Psi}_m^{\Delta t^{[j_k]}_{\ell^{[j_k]}(k)}}(\mathbf{\Phi}^{[j_k]}_{\ell^{[j_k]}(k)}) \| \\
&= \| \mathbf{\Psi}_{m+1}^{\Delta t^{[j_k]}_{\ell^{[j_k]}(k)}}(\mathbf{\Phi}^{[j_k]}_{\ell^{[j_k]}(k)}) - \mathbf{\Psi}_m^{\Delta t^{[j_k]}_{\ell^{[j_k]}(k)}}(\mathbf{\Phi}^{[j_k]}_{\ell^{[j_k]}(k)}) + \mathbf{\Gamma}^{\Delta t^{[j_k]}_{\ell^{[j_k]}(k)}}(\mathbf{\Phi}^{[j_k]}_{\ell^{[j_k]}(k)}) - \mathbf{\Psi}_{m+1}^{\Delta t^{[j_k]}_{\ell^{[j_k]}(k)}}(\mathbf{\Phi}^{[j_k]}_{\ell^{[j_k]}(k)}) \| \\
&\approx \| \mathbf{\Psi}_{m+1}^{\Delta t^{[j_k]}_{\ell^{[j_k]}(k)}}(\mathbf{\Phi}^{[j_k]}_{\ell^{[j_k]}(k)}) - \mathbf{\Psi}_m^{\Delta t^{[j_k]}_{\ell^{[j_k]}(k)}}(\mathbf{\Phi}^{[j_k]}_{\ell^{[j_k]}(k)}) \|. \qquad (4.28)
\end{aligned}
$$

Similar to the synchronous methods, the accuracy of the estimate is good under the assumption that the component force vector is not changing much within the timestep.

4.2.3 Timestep Size Selection

Unlike in the case of single timestep methods, where timestep size selection is made before the numerical solution is calculated, timestep size selection is made after the numerical solution is calculated. To avoid backward time marching (see Eq. 4.7), timestep size selection is not made within the timestep. Instead, after the numerical solution is computed at time (t_{k+1}) and the component forcing vector $(\mathbf{F}^{[j_k]}_{k+1})$ is evaluated, the *local discretization error* contributed by the component (j_k), at time t_{k+1}, is estimated and accordingly the component timestep $(\Delta t^{[j_k]}_{\ell^{[j_k]}(k+1)})$ size selection is made according to the rules stated in Eq. 4.29.

$$
\Delta t^{[j_k]}_{\ell^{[j_k]}(k+1)} = \begin{cases}
0.5 \times \Delta t^{[j_k]}_{\ell^{[j_k]}(k)} & \text{When} \quad \left(\dfrac{0.5 \times \tau^{[j_k]}}{\epsilon^{[j_k]}\left(\Delta t^{[j_k]}_{\ell^{[j_k]}(k)}\right)} \right)^{\frac{1}{m+1}} < 0.5 \\
\Delta t^{[j_k]}_{\ell^{[j_k]}(k)} & \text{When} \quad 0.5 < \left(\dfrac{0.5 \times \tau^{[j_k]}}{\epsilon^{[j_k]}\left(\Delta t^{[j_k]}_{\ell^{[j_k]}(k)}\right)} \right)^{\frac{1}{m+1}} < 2.0 \\
2.0 \times \Delta t^{[j_k]}_{\ell^{[j_k]}(k)} & \text{When} \quad 2.0 < \left(\dfrac{0.5 \times \tau^{[j_k]}}{\epsilon^{[j_k]}\left(\Delta t^{[j_k]}_{\ell^{[j_k]}(k)}\right)} \right)^{\frac{1}{m+1}}
\end{cases} \qquad (4.29)
$$

In linear multistep methods, using a similar strategy as in (4.29) essentially results in the timestep being predominantly constant in phases and therefore the error estimator (4.28) would be accurate. Similarly, since component timesteps are chosen such that the *local discretization error* matches the specified tolerance, TSS strategy (4.29) results in component timestep being predominantly constant in phases leading to the error estimator (4.28) being accurate. In a way, with a proper choice force field splitting, TSS-MASM does not have to deal with issues related to modeling *stiff* systems while modeling them.

4.3 Numerical Studies and Discussion

In this section, we will study the performance of TSS strategy (see Section 4.2.3) that can be employed while using MASM for numerically solving ODEs. The performance of the above TSS-MASM method is tested against a sdof spring-mass system, as well as against a nonlinear aerosol condensation model. We have seen in Section 2.4, that splitting of the force field play a key role in determining the performance of MASM. We explore the same in the context of TSS-MASM, more precisely against force field splitting that satisfy *time scale splitting* criterion as well as the force field splitting that satisfy *asynchronous splitting* criteron. Spring-mass model is an autonomous single timescale system. It allows us to study the performance when splitting a single timescale or eigen component, *i.e time scale splitting*.

In case of aerosol dynamics problem, we simulate the evolution of *volume* of the aerosol particles as it interacts with water vapor in the atmosphere and grow *via*, condensation process. The scale at which each particle grows depend upon the size of the particle and as the size of the particle evolve, so does the scale at which it evolves. The evolution of aerosol particles have a weak interdependence and that is *via* water vapor concentration. As a result by choosing a different timestep to update the evolution of each aerosol particle, leads to a splitting that is very close to *asynchronous splitting*.

4.3.1 Linear mass spring system

Consider a single-degree-of-freedom linear spring-mass system, as shown in Figure 2.2. The governing equation are given by

$$\ddot{x} + k_t x = 0, \tag{4.30}$$

where $k_t = k_1 + k_2$. The split first order ODE form chosen is given by,

$$\dot{\mathbf{X}} = \begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & \frac{r_s}{1+r_s} \\ -(\frac{r_s}{1+r_s})k_t & 0 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix}}_{\mathbf{F}^{[1]}(\mathbf{X})} + \underbrace{\begin{bmatrix} 0 & \frac{1}{1+r_s} \\ -(\frac{1}{1+r_s})k_t & 0 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix}}_{\mathbf{F}^{[2]}(\mathbf{X})}, \tag{4.31}$$

We took initial conditions $x(0) = 0$ and $v(0) = 1$, parameter $k_t = 100$, and we chose the timestep for each component to be

$$\Delta t_0^{[p]} = \frac{2\pi h}{10}\sqrt{\frac{1}{k_t}}, \tag{4.32}$$

in terms of a parameter h.

Given (4.30) is an conservative system, split vector field (4.31) represent *time scale splitting* example. Performance of MASM with timestep adaptation studied in Section 4.2.3, ABM with and without a similar timestep adaptation discussed in Section 4.2.3. The only difference in timestep adaptation employed with ABM being, the timestep correction is made before the numerical solution is calculated, which has been the approach in *synchronous* methods. The cost of a simulation is given by $\delta_{\text{total}}/\delta$ (See Eq. 2.80) with $\gamma = 1.0$.

Each figure in the panel of figures shown in Fig. 2.3, shows the two-norm error of the system state at time $t = 2$ *seconds*, versus cost ($\delta_{\text{total}}/\delta$) curves, obtained using TSS-MASM and the same is compared with the curves obtained using TSS-ABM and ABM. Different figures in the panel are for $m = \{2, 3, 4, 5\}$. One can see that TSS-ABM is no more efficient than ABM for $m = \{2, 3, 4, 5\}$, which can be explained from the fact that the spring-

mass model is an autonomous system. TSS-MASM curves also found to be as efficient as TSS-ABM for $m = \{2, 3, 4, 5\}$.

This result is not surprising, since system components share the same eigenvalue. This results in TSS-MASM choosing the same timestep and therefore is equivalent to TSS-ABM. No computational gain can be achieved even when the cost ratios are skewed (*i.e* $\gamma < 1$), when the component timesteps chosen are the same.

4.3.2 Nonlinear aerosol condensation system

We now consider a simplified model of water condensation onto aerosol particles, which was also studied in Section 2.4.2. The full dynamics of this process are complicated and for simplicity of exposition we use a extremely simple model here, whose governing equations are given by (2.82 - 2.83). The initial population of aerosol particles and their distribution is given by (2.86), with initial water vapor concentration, $W(0) = 20$.

Aerosol condensation problem (2.82 - 2.83) was studied using MASM with timestep adaptation introduced in Section 4.2.3, *i.e* TSS-MASM. These results are compared with the results from MASM's *synchronous* version with (TSS-ABM) and without (ABM) a similar timestep adaptation discussed in Section 4.2.3. The only difference between timestep adaptation used in TSS-ABM is that the timestep is adapted before the numerical solution is calculated at each time.

All the simulations using TSS-ABM and ABM are initialized with constant timestep given by,

$$\Delta t_0 = .15 \times \frac{V_1}{|\dot{V}_1|} \tag{4.33}$$

where V_1 is the smallest aerosol particle in the initial populations chosen (2.86). However, in simulations using MASM, we split the system into the $n + 1$ components given by (2.82)

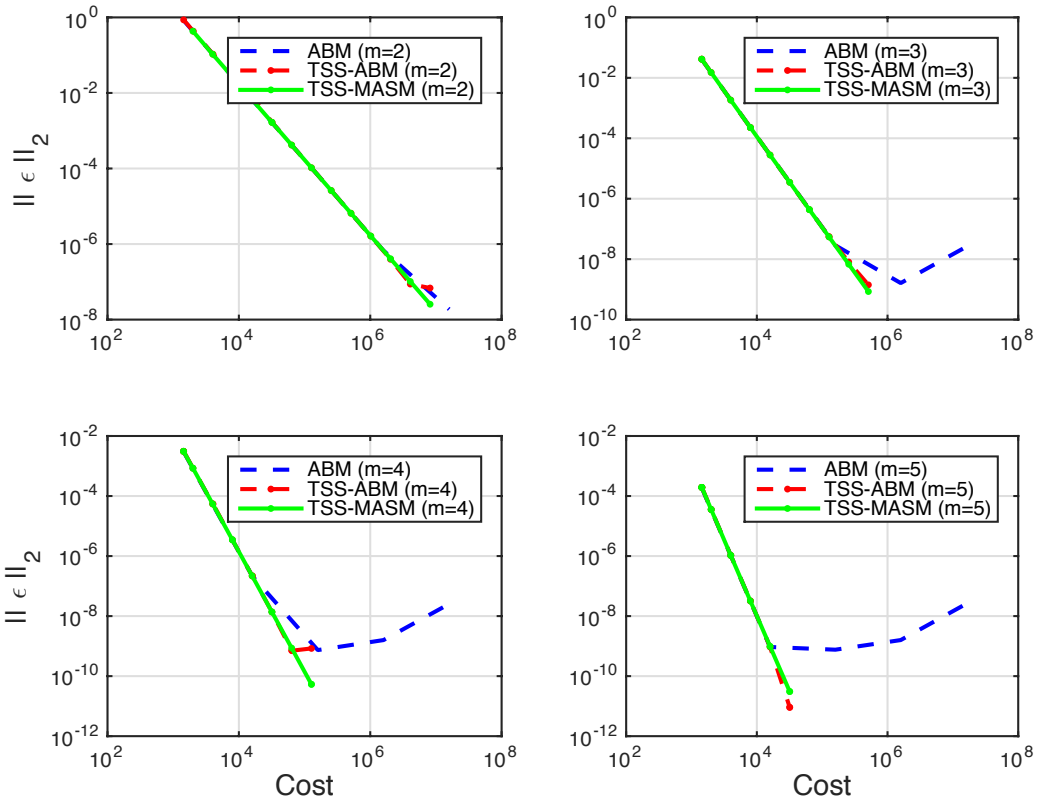

Figure 4.1: Error-versus-cost curves for the linear spring-mass system of Section 4.3.1 using asynchronous TSS-MASM integrator with force field splitting (4.31) and *synchronous* TSS-ABM integrators, with varying error tolerance τ. Error-versus-cost curves obtained using ABM are also shown for comparison. The cost is $\delta_{\text{total}}/\delta$ (2.80), while the error is the two-norm state error at the final time. We observe that the m-step methods are of order m in both the *synchronous*, with or without timestep adaptation, and asynchronous cases. *Synchronous* methods, with or without timestep adaptation have identical error-cost curves, owing to the fact that spring-mass system is an autonomous system. Asynchronous TSS-MASM also seems to be as efficient as synchronous methods, because of the time scale splitting (4.31) for the force vector field

and (2.83) and choose per-component timesteps

$$\Delta t_0^{[j]} = .15 \times \frac{V_1}{|\dot{V}_1|} \text{ for } j = 1, \ldots, n+1 \tag{4.34}$$

Error-versus-cost curves obtained using TSS-MASM, TSS-ABM and ABM are shown in Fig. 4.2 for $m = \{2, 3, 4, 5\}$. Given the aerosol model in Eq. 2.82 is nonlinear in nature, TSS-ABM, as expected, is more efficient than ABM. Unlike the spring-mass system studied in Section 4.3.1, which is a single timescale system, aerosol model (2.82 - 2.83) is multiple-time-scale system, with each aerosol particle evolving at different timescale. Given that the evolution of an aerosol particle does not directly depend upon dynamics of other particles, but indirectly through water vapor concentration, its is also a loosely coupled model. In simulations using MASM, the system split as in Eq. 4.34. Such a choice of system split falls very close to *asynchronous splitting*, with the system components being loosely coupled. As a result, the TSS-MASM is found to be more efficient than TSS-ABM, even without any favorable cost distribution of component system evaluations.

Top figure in Fig. 4.3 shows the evolution of aerosol particles with time, including the water vapor concentration. Bottom figure in Fig. 4.3 shows the evolution of different timesteps tracking the evolution of different aerosol particles under TSS-MASM. The bottom plot in Fig. 4.3 also the time variation of the timestep chosen in TSS-ABM and ABM simulations. Given the spread in the component timestep chosen in bottom plot in Fig. 4.3, it clearly shows the potential computational gain that could be achieved with favorably skewed cost ratios of component force vector evaluations.

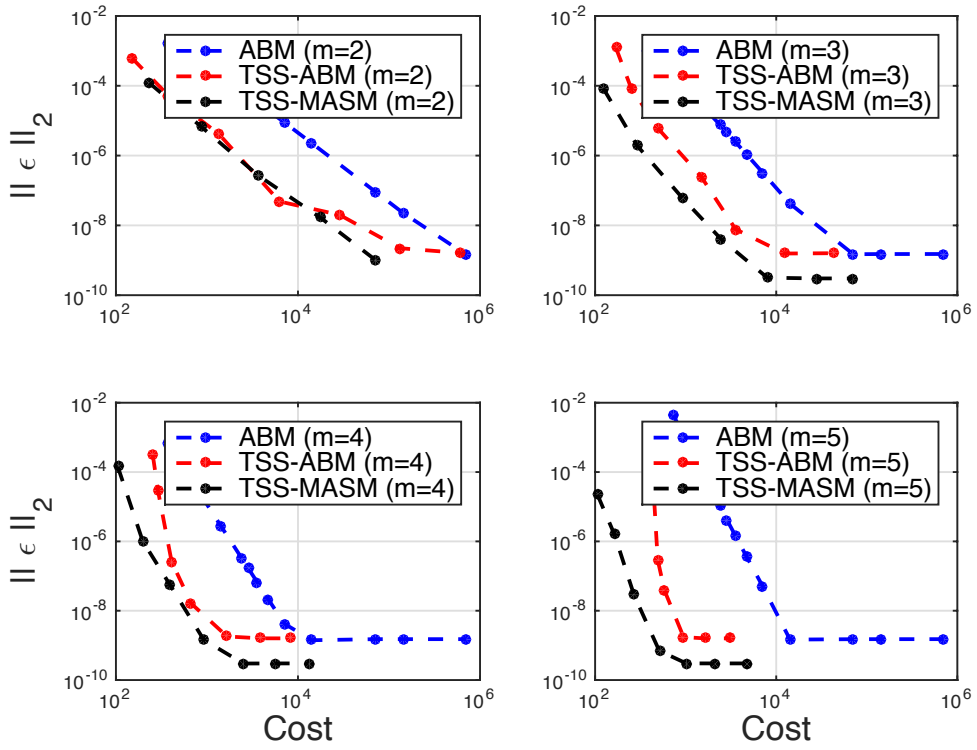

Figure 4.2: Error-versus-cost curves for the nonlinear aerosol condensation system of Section 4.3.2 using the asynchronous TSS-MASM and *synchronous* ABM and TSS-ABM integrators, with varying cost. The cost is proportional to the total number of component function evaluations, while the error is the sup-norm state error at the final time. We see that the m-step methods are of order m, with or without timestep adaptation as discussed in Section 4.2.1. Nonlinear aerosol condensation model being a non-autonomous system, ABM show significant gain in efficiency by employing timestep adaptation similar to the one introduced in Section 4.2.3. As the aerosol model is a sparsely-coupled system with several timescales, the TSS-MASM is found to be more efficient than its *synchronous* version, TSS-ABM. With skewed cost distribution of component force vector evaluations, the efficiency gain can be much larger when using TSS-MASM.

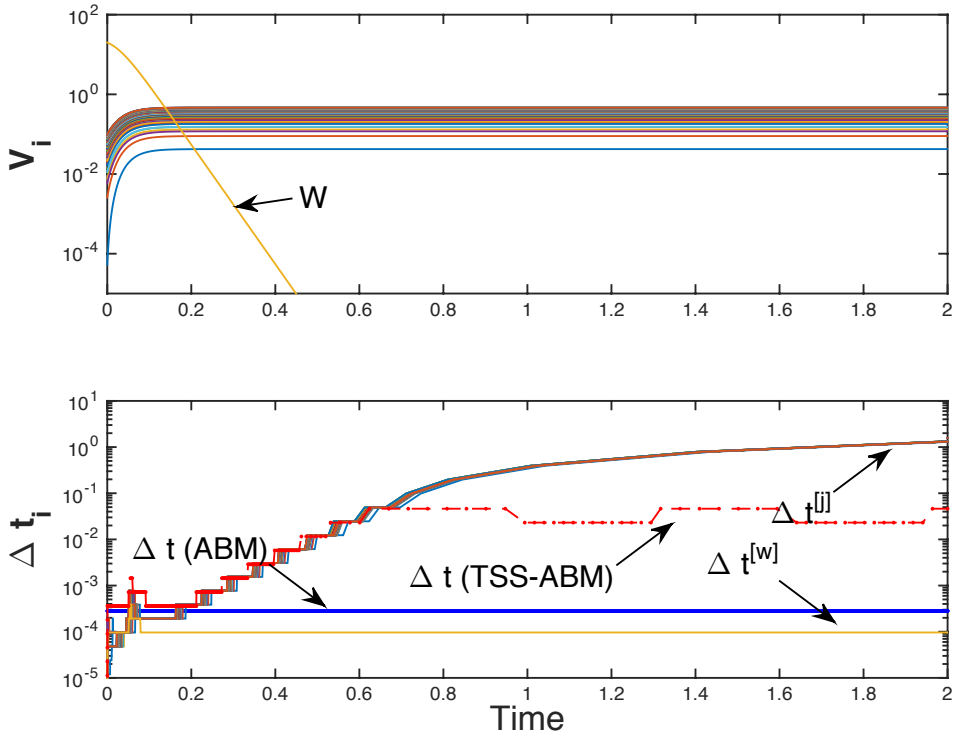

Figure 4.3: (Top figure) Shows the evolution of aerosol particles with time because of condensation studied in Section 4.3.2. Whose governing equations are given by Eqs. (2.82, 2.83). (Bottom figure) Shows the evolution of timesteps chosen for capturing the dynamics of different aerosol particles against time when modeling Eqs. (2.82, 2.83) using MASM with timestep selection strategy introduced in Section 4.2.3. The plot also shows the timestep evolution curves when using *synchronous* integrators such as TSS-ABM and ABM. As expected each timesteps used in TSS-MASM evolve with changing dynamics of each aerosol particle, whereas the timestep in TSS-ABM simulation evolve with overall dynamics. The timesteps chosen for each aerosol particle under asynchronous method are wide spread and show great potential to exploit any favorable cost distribution of component force vector evaluations.

Chapter 5

Conclusions

We have presented a new family of asynchronous multistep methods, the Multistep Asynchronous Splitting Methods (MASM), which are able to use a different timestep for each component of an additively-split ODE. These methods generalize both classical multistep integrators for ODEs [28] as well as Asynchronous Splitting Methods (ASM) [45]. We focused on MASM integrators of Adams type, with only the most recent system state being used in the update rule. We have shown that the MASM solution converges provided the ratio of timesteps chosen are bounded from above. We also proved that an m-step method of this type can have order m global error in the numerical solution of a sufficiently regular ODE system. This results was also demonstrated numerically on both linear and nonlinear systems.

The MASM integrators are generally more expensive per-step than a classic linear multistep method, as the method parameters $\langle \boldsymbol{\beta}_k^{[j]} \rangle_i$ must be solved for each component at every step with (2.29). Coupling between the split force fields play an important role in determining the performance of MASM. For strongly-coupled split vector fields, such as the spring-mass problem of Section 2.4.1, it is unlikely that MASM integrators will be superior to synchronous multistep methods. For sparsely-coupled split vector fields, however, such as the aerosol condensation problem of Section 2.4.2, we saw that MASM integrators can be much more efficient than the corresponding synchronous method.

We have also studied the stability of MASM [27] and have shown that MASM are stable in the Luxumberg [36] sense if the ratio of the largest and smallest timestep used is bounded from above, i.e., $(\max_{j,k} \Delta t_k^{[j]} / \min_{j,k} \Delta t_k^{[j]}) \leq Const.$ The stability criterion also strongly

depends upon on the splitting of the force field. In case of diagonalizable linear systems, any force field splitting can be classified into *asynchronous splitting*, where each eigen-component is lumped with one of the component force fields, and *time scale splitting*, where an eigen-component is split between two or more component force fields. We have shown that in the case of asynchronous splitting, $\mu_j (= \Delta t^{[j]} \times \lambda_{max})$ connected with each component force field, which is the product of the component timestep $\Delta t^{[*]}$ and the largest eigenvalue of system's Jacobian λ_{max}, need to satisfy the same stability criterion as its synchronous counterpart.

In the case of time scale splitting, we studied the stability of the method numerically using a single-degree-of-freedom spring-mass-damper system. We also restricted our study to 2-component analysis, where an eigen-component is shared between two component force fields. It was found that the timestep that determines stability is determined by the ratio of the split force vector norm and the ratio of timesteps, for conservative systems. In non-conservative systems, the timestep that determines stability is completely determined by the ratio of split force vector norms. As a result, when the fraction of each eigen-component of the force vector that is updated by smaller timestep is large enough, time scale splitting is equivalent to asynchronous splitting. As in asynchronous splitting, the timestep that determines stability in time scale splitting, approximately needs to satisfy the same stability criterion as its synchronous counterpart.

We also developed an adaptive timestep selection (TSS) strategy that can be used with MASM. We have shown that, in the case of MASM, the time adaptation strategy should be based on choosing the largest component timestep such that the contribution from all the component force fields to local discretization error are equal and together is below the specified tolerance. The timestep selection strategy was tested against single-degree-of-freedom spring-mass systems, as well as the nonlinear aerosol condensation problem. The performance of MASM with TSS strongly depends upon the splitting of the force vector field. For the spring-mass-damper model, where the force field splitting is time scale splitting, MASM based timestep selection strategy is as efficient as the similar timestep selection

strategy using synchronous ABM. However, in the case of nonlinear aerosol condensation problem, where different timesteps are used for the evolution of each aerosol particle, MASM with TSS is found to be more efficient than its synchronous counterpart. The efficiency gain can be further extended, when the cost distribution of split force vector evaluations are favorably skewed.

Appendix A

Theorem applicable to variable grid multistep methods

For the benefit of the readers, here we have reproduced a theorem from [70] whose result is used in *Theorem* 4.

Theorem 5. *Consider the linear difference equation*

$$\mathbf{Y}_{k+1} = \mathbf{Y}_k + \Delta t_k \left[\left(\sum_{i=0}^{\tilde{K}-1} \hat{\beta}_{k,i} \mathbf{Y}_{k-i} \right) + \mathbf{\Lambda}_{k+1} \right]. \tag{A.1}$$

Let B, $\mathbf{\Lambda}$, Y be non-negative constants such that,

$$\sum_{i=0}^{\tilde{K}-1} |\hat{\beta}_{k,i}| \leq B \text{ for } k \geq 0 \tag{A.2}$$

$$|\mathbf{\Lambda}_k| \leq \mathbf{\Lambda} \text{ for all } k \geq 0 \tag{A.3}$$

$$\mathbf{Y}_k \leq \mathsf{Y} \text{ for all } 0 \leq k \leq \tilde{K} - 1. \tag{A.4}$$

Then

$$|\mathbf{Y}_k| \leq e^{B(t_k - t_{\tilde{K}-1})} \left[\mathsf{Y} + \mathbf{\Lambda} \left(t_k - t_{\tilde{K}-1} \right) \right].$$

Proof.

$$\mathbf{Y}_{k+1} = \mathbf{Y}_k + \Delta t_k \left[\left(\sum_{i=0}^{\tilde{K}-1} \hat{\beta}_{k,i} \mathbf{Y}_{k-i} \right) + \mathbf{\Lambda}_{k+1} \right]. \tag{A.5}$$

Let $\{\mathbf{W}_i\}$ such that,

$$\mathbf{W}_0 = \mathbf{Y}_0 \tag{A.6}$$

$$\mathbf{W}_i = \max\left\{|\mathbf{Y}_i|, \mathbf{W}_{i-1}\right\}\ i = 1, 2, \ldots \tag{A.7}$$

$$|\mathbf{Y}_{k+1}| \leq \mathbf{W}_k + \Delta t_k \sum_{i=0}^{\tilde{K}-1} |\hat{\beta}_{k,i}| \mathbf{W}_{k-i} + \Delta t_k \mathbf{\Lambda}_{k+1}, \tag{A.8}$$

$$\leq \left(1 + \Delta t_k B\right) \mathbf{W}_k + \Delta t_k \mathbf{\Lambda}. \tag{A.9}$$

It obvious that,

$$\mathbf{W}_k \leq \left(1 + \Delta t_k B\right) \mathbf{W}_k + \Delta t_k \mathbf{\Lambda}. \tag{A.10}$$

Since $\mathbf{W}_{k+1} = \max\left(|\mathbf{Y}_{k+1}|, \mathbf{W}_k\right)$,

$$\mathbf{W}_{k+1} \leq \left(1 + \Delta t_k B\right) \mathbf{W}_k + \Delta t_k \mathbf{\Lambda}, \tag{A.11}$$

$$\leq e^{(\Delta t_k B)} \mathbf{W}_k + \Delta t_k \mathbf{\Lambda}, \tag{A.12}$$

It can be shown that,

$$\mathbf{W}_{k+1} \leq e^{B\left(t_{k+1} - t_{\tilde{K}-1}\right)} \left(\Upsilon + \mathbf{\Lambda}\left(t_{k+1} - t_{\tilde{K}-1}\right)\right). \tag{A.13}$$

□

References

[1] R. Abedi, S. H. Chung, J. Erickson, Y. Fan, M. Garland, D. Guoy, R. B. Haber, J. Sullivan, and Y. Zhou. Space- time meshing with adaptive refinement and coarsening. In *Proceedings of the 20th Annual ACM Symposium on Computational Geometry, Brooklyn, NY, USA*, Association for Computing Machinery, pages 300–309, 2004.

[2] R. Abedi, M. A. Hawker, R. B. Haber, and K. Matous. An adaptive spacetime discontinuous galerkin method for cohesive models of elastodynamic fracture. *International Journal for Numerical Methods in Engineering*, 81:1207, 2010.

[3] R. Abedi, B. Petracovici, and R. B. Haber. A spacetime discontinuous Galerkin method for linearized elastodynamics with element-wise momentum balance. *Comput. Methods Appl. Mech. Engrg.*, 195:3247, 2006.

[4] B. Andreas and G. Michael. Developments in multirating for coupled systems. *J. Applied Mathematics Mechanics*, 81:53, 2001.

[5] E. Barth and T. Schlick. Extrapolation versus impulse in multiple-timestepping schemes. ii. linear analysis and applications to newtonian and langevin dynamics. *J. Chem. Phys.*, 109:1633, 1998.

[6] E. Barth and T. Schlick. Overcoming stability limitations in biomolecular dynamics. i. combining force splitting via extrapolation with langevin dynamics in ln. *J. Chem. Phys.*, 109:1617, 1998.

[7] J. J. Biesiadecki and R. D. Skeel. Dangers of Multiple Time Step Methods. *J. Comp. Phys.*, 109:318–328, 1993.

[8] Manuel Calvo and Jesús Vigo-Aguiar. A note on the step size selection in adams multistep methods. *Numerical Algorithms*, 27(4):359–366, 2001.

[9] Oriol Colomes, Santiago Badia, Ramon Codina, and Javier Principe. Assessment of variational multiscale models for the large eddy simulation of turbulent incompressible flows. *Comput. Methods Appl. Mech. Engrg.*, 285:32 – 63, 2015.

[10] E. M. Constantinescu and A. Sandu. Multirate timestepping methods for hyperbolic conservation laws. *J. Sci. Comput.*, 33:239, 2007.

[11] M. Crouzeix and F. J. Lisbona. The convergence of variable-stepsize, variable-formula, multistep methods. *SIAM Journal of Numerical Analysis*, 21:512, 1984.

[12] W. J. T. Daniel. Analysis and implementation of a new constant acceleration subcycling algorithm. *International Journal of Numerical Methods in Engineering*, 40:2841, 1997.

[13] W. J. T. Daniel. The subcycled Newmark algorithm. *Computational Mechanics*, 20:272, 1997.

[14] W. J. T. Daniel. A study of the stability of subcycling algorithms in structural dynamics. *Computer Methods in Applied Mechanics and Engineering*, 156:1, 1998.

[15] W. J. T. Daniel. A partial velocity approach to subcycling structural dynamics. *Computer Methods in Applied Mechanics and Engineering*, 192:375, 2003.

[16] J. Erickson, D. Guoy, A. Ungor, and J. Sullivan. Building spacetime meshes over arbitrary spatial domains. In *Proceedings of the 11th International Meshing Roundtable, Ithaca, NY, U.S.A*, pages 391–402, 2002.

[17] W. Fong, E. Darve, and A Lew. Stability of asynchronous variational integrators. *Journal of Computational Physics*, 227:8367, 2008.

[18] B. Garcia-Archilla, J. M. Sanz-Serna, and R. D. Skeel. Long-time-step methods for oscillatory differential equations. *SIAM J. Sci. Comput.*, 20:930–963, 1998.

[19] C. W. Gear and I. G Kevrekidis. Projective methods for stiff differential equations: problems with gaps in their eigenvalue spectrum. *SIAM J. Sci. Comp.*, 24:1091, 2003.

[20] C. W. Gear and K. W. Tu. The effect of variable mesh size on the stability of multistep methods. *SIAM Journal Numerical Analysis*, 11:1025, 1974.

[21] C. W. Gear and D. R. Wells. Multirate linear multistep methods. *BIT*, 24:484, 1984.

[22] C William Gear. *Numerical initial value problems in ordinary differential equations*. Prentice Hall PTR, 1971.

[23] A. Gravouil and A. Combescure. Multi-time-step and two-scale domain decomposition method for non-linear structural dynamics. *International Journal of Numerical Methods in Engineering*, 58:1545, 2003.

[24] H. Grübmuller, H. Heller, A. Windemuth, and K. Schulten. Overcoming stability limitations in biomolecular dynamics. i. combining force splitting via extrapolation with langevin dynamics in ln. *Mol. Sim.*, 6:121–142, 1990.

[25] R. D. Grigorieff. Stability of multistep-methods on variable grids. *Numerische Mathematik*, 42:359, 1983.

[26] H. Grubmüller, H. Heller, A. Windemuth, and K. Schulten. Generalised Verlet algorithm for efficient molecular dynamics simulations with long-range interactions. *Mol. Sim.*, 6:121, 1991.

[27] P. K. Gudla, M. Shin, and M. West. High-order Multistep Asynchronous Splitting Methods (MASM) for MULTISCALE DYNAMICS. *ASME J. Comp. Nonlinear Dynamics*, under review.

[28] E. Hairer, S. P. Norsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer-Verlag, second edition, 1993.

[29] E. Hairer, S. P. Norsett, and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential Algebraic Problems*. Springer-Verlag, second revised edition, 1996.

[30] G. Han, Y. Deng, J. Glimm, and G. Martyna. Error and timing analysis of multiple time-step integration methods for molecular dynamics. *Computer Physics Communications*, 176:271, 2007.

[31] A. Hayli. Le problème des N corps dans un champ extérieur. Application à l'évolution dynamique des amas ouverts. I. *Bull. Astronomique*, 2:67, 1967.

[32] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, second edition, 1961.

[33] T. J. R. Hughes, T. Belytschko, and W. Liu. Convergence of an element partitioned subcycling algorithm for the semi-discrete heat equation. *Numerical Methods for Partial Differential Equations*, 3:131, 1987.

[34] T.J.R. Hughes and G.M. Hulbert. Space-time finite element methods for elastodynamics: Formulations and error estimates. *Comput. Methods Appl. Mech. Engrg.*, 66:339, 1988.

[35] G.M. Hulbert and T.R.J. Hughes. Space-time finite element methods for second order hyperbolic equations. *Comput. Methods Appl. Mech. Engrg.*, 84:327, 1990.

[36] T. E. Hull and W. A. J. Luxumberg. Numerical methods and existence theorems for ordinary differential equations. *Numerische Mathematik*, 2:30, 1960.

[37] A. Iserles. *A First Course in the Numerical Analysis of Differential Equations*. Cambridge University Press, 1996.

[38] T. Kato and T. Kataoka. Circuit analysis by a new multirate method. *Electr. Eng. Jpn.*, 126:55, 1999.

[39] B. Kraczek, S. T. Miller, R. B. Haber, and D. D. Johnson. Adaptive spacetime method using riemann jump conditions for coupled atomistic-continuum dynamics. *J. Computational Physics*, 229:2061, 2010.

[40] Fred T. Krogh. Changing stepsize in the integration of differential equations using modified divided differences. In *Bulletin of the American Astronomical Society*, volume 4, pages 420–421, 1972.

[41] Fred T. Krogh. Algorithms for changing the step size. *SIAM Journal on Numerical Analysis*, 10(5):949–965, 1973.

[42] Fred T. Krogh. Changing stepsize in the integration of differential equations using modified divided differences. In *Proceedings of the Conference on the Numerical Solution of Ordinary Differential Equations*, pages 22–71. Springer, 1974.

[43] Fred T. Krogh. Stepsize selection for ordinary differential equations. *ACM Trans. Math. Softw.*, 37(2):1–11, 4 2010.

[44] A. Lew, J. E. Marsden, M. Ortiz, and M. West. Asynchronous variational integrators. *Archive for Rational Mechanics and Analysis*, 167:85, 2003.

[45] A. Lew, J. E. Marsden, M. Ortiz, and M. West. Variational time integrators. *International Journal for Numerical Methods in Engineering*, 60:153, 2004.

[46] R. I. McLachlan and G. R. W. Quispel. Splitting methods. *Acta Numerica*, 11:341–434, 2002.

[47] P. Minary, M. Tuckerman, and G. Martyna. Long time molecular dynamics for enhanced conformational sampling in biomolecular systems. *Phys. Rev. Lett.*, 93:150201, 2004.

[48] M. O. Neal and T. Belytschko. Explicit-explicit subcycling with non-integer time step ratios for structural dynamic systems. *Computers and Structures*, 6:871, 1989.

[49] L. R. Petzold, L. O. Jay, and J. Yen. Numerical solution of highly oscillatory ordinary differential equations. *Act Numerica*, 6:437, 1997.

[50] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 3rd edition, 2007.

[51] A. Sandu and E. M. Constantinescu. Multirate explicit adams methods for time integration of conservation laws. *J. Sci. Comput.*, 38:229, 2009.

[52] A. Sandu and T. Schlick. Masking Resonance Artifacts in Force-Splitting Methods for Biomolecular Simulations by Extrapolative Langevin Dynamics. *J. Comp. Phys.*, 151:74–113, 1999.

[53] J. M. Sanz-Serna and M. P. Calvo. *Numerical Hamiltonian Problems*. Chapman and Hall, first edition, 1994.

[54] T. Schlick, M. Mandziuk, R. Skeel, and K. Srinivas. Nonlinear resonance artifacts in molecular dynamics simulations. *J. Comp. Phys.*, 139:1, 1998.

[55] J. H. Seinfeld and S. N. Pandis. *Atmospheric chemistry and physics*. Wiley, 2006.

[56] Lawrence F. Shampine. The step sizes used by one-step codes for odes. *Applied Numerical Mathematics*, 1(1):95–106, 1985.

[57] LAWRENCE F. Shampine. Variable order adams codes. *Computers & Mathematics with Applications*, 44(5):749–761, 2002.

[58] Lawrence F Shampine and Marilyn Kay Gordon. *Computer solution of ordinary differential equations: the initial value problem*. WH Freeman San Francisco, 1975.

[59] Lawrence F. Shampine and Mark W. Reichelt. The matlab ode suite. *SIAM journal on scientific computing*, 18(1):1–22, 1997.

[60] Lawrence F. Shampine and Mark W. Reichelt. The matlab ode suite. *SIAM journal on scientific computing*, 18(1):1–22, 1997.

[61] Minyong Shin and Matthew West. High-order multistep asynchronous splitting integrators (masi). In *Proceedings of 5th European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2008), June 30 - July 5 (2008), Venice, Italy*, 2008.

[62] P. Simolinski and Y. S. Wu. An implicit multi-time step integration method for structural dynamics problems. *Computers Mechanics*, 22:337, 1998.

[63] R. Skeel, G. Zhang, and T. Schlick. A family of symplectic integrators stability, accuracy and molecular dynamics applications. *SIAM J. Sci. Comput.*, 18:203, 1997.

[64] Robert D. Skeel. Construction of variable-stepsize multistep formulas. *MATHEMATICS of computation*, 47(176):503–510, 1986.

[65] W. B. Streett, D. Tildesley, and G. Saville. Multiple time step methods in molecular dynamics. *Mol. Phys.*, 35:639, 1978.

[66] W. B. Streett and D. J. Tildesley. Multiple time-step methods in molecular dynamics. *Mol. Phys.*, 35:639–648, 1978.

[67] F. C. Tseng. *Multibody dynamics simulation in network-distributed environments*. Ph.D Dissertation, University of Michigan, 2000.

[68] F. C. Tseng and G. M. Hulbert. A gluing algorithm for network-distributed dynamics simulation. *Multibody System Dynamics*, 6:377, 2001.

[69] F. C. Tseng, Z. D. Ma, and G. M. Hulbert. Efficient numerical solution of constrained multibody dynamics systems. *Computer Methods in Applied Mechanics and Engineering*, 192:439, 2003.

[70] K. W. Tu. Stability and convergence of general multistep and multivalue methods with variable step size. In *Doctoral Thesis, Department of Computer Science Rep. 526, University of Illinois at Urbana-Champaign, IL, U.S.A*, 1972.

[71] M. Tuckerman, B. J. Berne, and G. J. Martyna. Reversible multiple time scale molecular dynamics. *J. Chem. Phys.*, 97:1990, 1992.

[72] J. Z. Wang, Z. D. Ma, and G. M. Hulbert. A gluing algorithm for distributed simulation of multibody systems. *Nonlinear Dynamics*, 34:159, 2003.

[73] D. M. York, T. A. Darden, and L. G. Pedersen. The effect of long-range electrostatic interactions in simulations of macromolecular crystals: A comparison of the Ewald and truncated list methods. *Journal of Chemical Physics*, 99:8345, 1993.

[74] K. M. Zhang and A. S. Wexler. An asynchronous time-stepping (ATS) integrator for atmospheric applications: Aerosol dynamics. *Atmos. Environ.*, 40:4574, 2006.