Application of the GSSSS Family of Algorithms to the Natural Index 3 Differential-Algebraic Equations of Multibody Dynamics

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Andrew John Hoitink

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

Dr. K. Tamma

July, 2011

Acknowledgements

I would like to thank Prof. K. K. Tamma and Dr. Xiangmin Zhou for all their guidance and help throughout my graduate school career. This work would not have been possible without their tremendous focus and advice.

I would also like to thank Siti Masuri and Masao Shimada for their collaboration, advice, and friendship along the way.

Dedication

To my loving parents and wife

Abstract

Growing interest in the simulation of constrained multibody systems has prompted the development of a variety of time integration methods for the differential-algebraic equations (DAEs) inherent to these systems. The vast majority of these methods require reformulation of the natural index 3 equations of motion leading to additional computational cost and the need to control drift phenomena, citing instability of direct index 3 methods. This research sought within the generalized single step single solve (GSSSS) family of algorithms an algorithm and framework capable of overcoming the instability and problems encountered by previous researchers. A precise understanding of the equation of motion time level concept as well as novel methods for extending linear parent algorithms to nonlinear dynamic applications enabled a depth of search unique to the area. In the end, an algorithmic framework is identified which overcomes previous limitations and is capable of providing stable, robust, and accurate integration of index 3 DAEs for both rigid and rigid/flexible multibody dynamics applications.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

- Chapter 1 introduces the goals and motivation of the research presented in this thesis.
- Chapter 2 provides background information and an overview of previous research about the GSSSS family of algorithms and its extension to nonlinear problems.
- In Chapter 3 the possible methods of extending these algorithms to solve differential-algebraic equations are described.
- Chapter 4 illustrates the results of the numerical examples for which these methods were tested.
- Chapter 5 presents a final discussion of the analyses presented in the thesis and summarizes the conclusions drawn within.

1.1 Research Objectives and Motivation

Of interest in this research was the development and design of robust time integration algorithms for two classes of applications: 1) elastodynamics, or that referred to as structural dynamics in the literature where no constraints are present, and

2) multibody dynamics (both rigid and flexible) where constraints are inherent in the dynamic system. Both these classes of applications are mostly dominated by low frequency effects, unlike the class of applications in wave propagation, molecular dynamics, etc., where high frequency is equally important and dominant. In this context, for the former applications mostly implicit time integration methods are employed, while in the latter mostly explicit methods of integration of the dynamic equations of motion are preferred. This work primarily focuses on low frequency dynamic applications that are frequently encountered in vibrations and dynamics of mechanical, aerospace, and civil engineering structures, which are frequently referred to as inertial problems. The primary motivation of this research for these applications is the following:

1) Analyze the key restrictions in the current methodology used in codes to simulate fully rigid systems (kinematics) as well as codes used to simulate coupled rigid and flexible systems.

2) Leverage the general nature of the GSSSS family of algorithms and recently developed understanding of its extension to nonlinear applications to search for algorithms which may be capable of overcoming these existing restrictions.

3) Develop a framework with computationally attractive features which allows direct coupling of rigid and flexible bodies for efficient combined multibody system simulation.

Unlike elastodynamics which mostly deals with low frequency dominated applications where structural elements in the system do not have any constraints, the goal within the area of multibody dynamics is to describe the dynamic behavior of systems interconnected through constraint equations. These bodies, which may be treated as rigid or flexible bodies, and are connected through joints which impose additional constraints to the standard dynamic system. In addition to the constraints imposed by joints connecting multiple bodies, there may also be geometric constraints such as analytically rigid components or predefined trajectories. These constraints transform the standard set of ordinary differential equations (ODEs) into a set of differential-algebraic equations (DAEs). The

ability to simulate the motion and deformation of such systems is of significant interest to the engineering community. Problems such as vehicle dynamics, analysis of wind turbines, or robot dynamics are just a few examples which fall within the area of multibody dynamics.

Frequently in the development of models for this class of problems it is beneficial to be able to consider some of the bodies as rigid while allowing others to undergo deformation. These rigid bodies simplify the model in that they are able to remove the deformation degrees of freedom and are more efficient to integrate. There are many commercially available codes which are currently used for simulation of rigid multibody systems such as ADAMS and DADS. These codes use "stiff" solvers to integrate the equation of motion for the system of rigid bodies but are mostly unable to consider deformation directly. Currently those who wish to simulate a system which has both rigid and flexible bodies must resort to two options: 1) first solve the system as if it were rigid body motion, then transfer the reaction forces at every time step to a code which can solve for the (pseudo) static deformations, or 2) utilize recently developed codes for coupled rigid/flexible which depend almost exclusively on explicit time integration methods. The process of first solving the rigid body dynamics and then transferring the inertial forces to a code such as NASTRAN which can solve for deformation and stresses is obviously quite time consuming and inefficient. As codes intended to solve structural dynamics problems most frequently utilize implicit time integration schemes, the need to turn to explicit or specialized schemes when coupling rigid and flexible bodies is certainly undesirable. A computational framework capable of efficiently and robustly simulating systems of both rigid and flexible bodies would be essentially the "holy grail" for the community. As such, investigation of the key restrictions and development of techniques to overcome these restrictions was one of the main objectives of this research.

Most multibody systems of practical interest involve very large displacements and rotations. As such, even for very simple linear material models these large

displacements make the problem highly geometrically nonlinear. This nonlinearity is problematic in that it greatly complicates integrating the spatially discrete equations of motion. Algorithms which for linear systems can be proven to be unconditionally stable are no longer guaranteed to provide a solution in the nonlinear case. Recently, research on the extension of the generalized single step single solve (GSSSS) family of implicit time integration algorithms has provided new insight into robust simulation of nonlinear elastodynamic systems. Concepts such as the normalized weighted residual procedure and the importance of consistency in the equation of motion time level enable the possibility of new insight into the development of methods which may overcome the restrictions from which existing methodologies suffer. It was these recent developments which fostered optimism that the GSSSS family may contain algorithms capable of overcoming the numerical difficulties encountered in the realm of highly nonlinear constrained dynamic simulations. The general nature of the GSSSS algorithms and the deep understanding of implementation for nonlinear equations makes this research unique to the community in that the vast majority of other researchers in the area are simply reusing the handful of well-known algorithms used in linear dynamics whereas we have the unique benefit of testing algorithms which are almost entirely unknown.

Though spatial discretization of systems is not the primary concern of this research, it has significant impact on the resulting system of equations. In general, for structural dynamics applications, the standard Galerkin finite element is used to take a continuous system and discretize it into a mesh of nodes and elements. The outcome of this discretization is often that some elements within the mesh have significantly higher frequency oscillation than others. A system which has elements of widely varying frequencies is considered to be a "stiff" numerical system. Integration of such systems has been the topic of research for many years and has led to the development of algorithms which exhibit controllable numerical dissipation. This dissipation may be thought of as artificial dampening of high frequency modes of the system, which for structural dynamic applications are understood not to play a key role in the solution. Algorithms which involve

controllable numerical dissipation result in a non-physical damping of total system energy. For this reason the current research set out to determine the applications for which controllable numerical dissipation was not necessary to foster a numerical solution, though the general nature of the GSSSS algorithms always includes the capability.

As it stands to-date, there are two main families of algorithms which fall under the umbrella of non-dissipative linear multistep (LMS) methods: Symplectic algorithms and algorithms which conserve energy and momentum exactly. Symplectic algorithms are characterized by the ability to conserve phase space volume which is very desirable in areas such as molecular dynamics and gravitational problems. In terms of total system energy however, symplectic algorithms provide oscillatory (although bounded) energy for conservative systems for a fixed time step. On the other hand, energy and momentum conserving methods (EMM) are able to conserve total energy, linear momentum, and angular momentum exactly over a time step but do not perfectly preserve phase space volume. Implementation aspects for symplectic algorithms are fairly well understood as the algorithm itself is not directly dependent on the material model. Conversely, implementation of EMM algorithms is still an active research area. Complexities arise from nonlinear material models as well as problems related to the overall stiffness of the system. The key desirable quality of an EMM algorithm is that one is able to claim unconditional stability in terms of system energy, even for highly nonlinear problems. Recent research has shown each of these properties (symplectic and EMM) may be derived for structural dynamic simulations by careful extension of the parent linear time integration algorithm to nonlinear applications, each with its own benefits and drawbacks. The current research focuses mainly on the energy conserving algorithms, renowned for their stable solution of stiff problems.

1.2 Overview of Existing Methods Versus Current Research

We intend to focus our attention on modeling and simulation of structural and multibody systems consisting of both rigid and flexible bodies as opposed to only rigid or only flexible bodies. The difference between structural and multibody systems is the presence of constraint equations in the formulation of the latter. These constraints equations can impose a wide variety of connections as well as analytically rigid elements. An example multibody system can be seen in Fig. 1.1 .

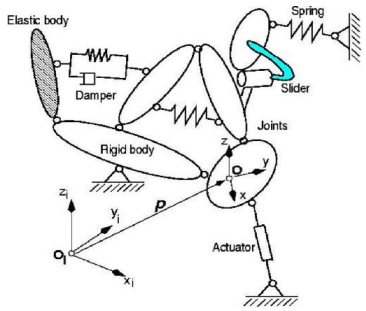


Figure 1.1: Multibody Constrained System from [1]

The concept of a rigid body, although somewhat imaginary (there is no perfectly rigid body), can be very useful if the deformations are small enough so as to not effect the gross motion of the body. Modeling a body as rigid as compared to flexible can lead to considerable computational savings. On the other hand some situations warrant the modeling of certain bodies as flexible due to significant deformations or the desire to analyze stresses within the body. Thus a technique which has the ability to simulate both rigid and flexible bodies at the same time provides users with means to vary the degree of fidelity of a model and also leads to computationally efficient modeling of joints which are assumed to be rigid, as opposed to modeling them as flexible which requires modeling of contact. Thus for a preliminary design where computational time is a premium,

a user may choose to model most of the members of the system as rigid. But in detailed design stages, more and more members can be modeled as flexible in order to accurately capture the physics of the problem. Popular multibody software like DADS and ADAMS feature efficient simulation of systems composed of rigid bodies only. These software handle the case of flexible bodies by interfacing with finite element software to calculate the mode shapes of deformation. This current research however attempts to develop a methodology that allows the simulation of multibody systems with rigid and flexible bodies simultaneously.

1.3 Literature Overview: Time Discretization

Of primary interest and the focus in this section are the numerical aspects of time integration of the equations of motion for holonomic-scleronomic systems that are routinely encountered in a wide class of problems in computational dynamics of particles, materials, and continuum structures such as elastodynamics. There exist various frameworks for developing computational formulations, namely, vector formalisms and scalar formalisms. The limitations for establishing equivalences of the strong and weak forms amongst the various frameworks (Newtonian, Lagrangian, Hamiltonian, and the Total Energy) are that the kinetic energy is quadratic in generalized velocities and the potential energy is not velocity dependent for N-body systems. For continuum dynamics applications we additionally restrict attention to the case when the kinetic energy is independent of the generalized coordinates (that is, the mass matrix is constant). With the exception of the Hamiltonian framework which yields a system of first order in time semi-discretized equations of motion, all the other frameworks give rise to a semi-discretized system of second order in time differential equations. Historically, most time integration methods have been derived in the sense of the single field form of representations for the equations of motion. In general, the design of computational algorithms for the integration of the equations of motion either pertain to a two-field (first order) form or a single field (second order) form of representation for the process of time

integration. Consequently, the former representation does not permit a spurious root (non-dissipative algorithm design). However, unless accelerations are not treated as a primary variables, for which case a spurious root does not exist, the latter naturally involves a spurious root that is associated with the underlying algorithmic amplification matrix and provides the ability to develop controllable algorithmic dissipation. This dissipation can be employed to control the amount of participation of the high frequency modes resulting from the numerical space discretization. In this section, we particularly focus attention to the class of LMS methods involving a single system of equations and a single solve within each time step as these are the most practical, economical, and commonly employed in commercial software settings. In particular we focus attention on the single field form of representations which gives rise to a 3-root system as the displacements, velocities, and accelerations are treated as the primary variables.

Traditionally, most of the classical time integration methods including linear multistep methods have been motivated from Taylor's series expansions for the approximation of the field variables such as displacement, velocity, and acceleration appearing in the space discretized equations of motion. Even though several of these methods have been developed to a certain degree of maturity and from various differing viewpoints, the geometric nature that the ordinary or partial differential systems possess has been ignored to an extent in constructing time integration algorithms in certain cases. Consequently, the time integration for highly nonlinear differential systems may become unstable and blows up for certain time steps.

To address some of these concerns, geometric mechanics stemming from Lagrangian mechanics which has been pioneered by Lie (1842-1899) appears to play an important role. In particular, to ensure that certain physical attributes are conserved/preserved in the algorithm design. It has been believed that the preservation of geometric nature of the differential system can give rise to long-term simulations, especially for nonlinear differential systems. Furthermore, preserving

the structure appears to provide more physically accurate time discretized algorithms [5] that are useful for a certain class of applications. Nonetheless, the goal of designing algorithms with a single system and single solve within each time step further enhances the efficiency of computations from a practical viewpoint. Hence, the focus of this research was restricted to such practical considerations.

Understanding the symmetry in physics is very important in developing time integration schemes that inherit certain desirable algorithmic attributes useful for certain classes of problems. Noether [6] first discovered that the Lagrangian has symmetry. Conservation of quantities such as energy, linear, and angular momentum results from the symmetry of the Lagrangian or the Hamiltonian. Hence, structure preserving algorithms have attracted some degree of attention, for example, the energy-momentum conserving and symplectic algorithms. Algorithms which are able to conserve fundamental physical quantities such as energy, linear momentum, and angular momentum have been the topic increasingly many research papers. Beyond the basic desire to exactly conserve quantities which we know from physics should be conserved, implicit EMM algorithms have the unique ability to claim unconditional stability of energy even for nonlinear problems. That is, regardless of the size of the time step taken the algorithm will maintain an exact total system energy. This is contrary to symplectic algorithms which often suffer from "jumps" in energy for time step sizes which are too large.

1.4 Literature Overview: Energy Conservation

Originally LaBudda and Greenspan [7, 8] developed EMM algorithms for particle dynamics problems. Greenspan then went on to propose an EMM algorithm which could be applied specifically to nonlinear dynamic problems as related to particle dynamic as well [9]. The first energy conserving algorithm for nonlinear structural dynamics was proposed by Hughes [10] using Lagrange multipliers in conjunction with the trapezoidal rule. This algorithm lacked momentum conservation properties and was later extended by Kuhl and Ramm [11] who introduced algorithmic

damping of high frequency response while conserving both energy and momentum. A more naturally derived EMM algorithm was developed later by Simo [12, 13] without the need for Lagrange multipliers. As this method was originally derived under a two-root system, the ability to impose controllable numerical dissipation to reduce high frequency participation in the solution is impossible. This has been shown in [11] to hinder the convergence of the required nonlinear iterations. The extension of Simo's EMM into a three-root system capable of imposing controllable numerical dissipation was done by Kuhl and Crisfield [14]. This method was able to exactly conserve energy and momentum with the dissipation parameter turned off as well as impose numerical dissipation when required (at the loss of conservation properties, of course). It is also restricted to linear material models.

Numerous computational implementations for EMM can be seen in [12, 13, 15, 16]. Some which are designed to for use with only simple linear material models and some which can be extended for use even with general hyperelastic nonlinear materials. Almost without exception it is observed that for highly nonlinear problems where the discretized system contains a wide range of frequencies (so called "stiff" systems) energy conserving algorithms foster solution stability. That is, for stiff systems with non-energy conserving algorithms it is often seen that the system energy will grow without bound leading to a very nonphysical solution as well as failure in nonlinear iteration convergence. Energy conserving algorithms on the other hand guarantee conservation of energy from one time step to the next, therefore it is impossible for the unbounded energy growth to occur.

The stabilizing nature of energy conservation did not go unnoticed by researchers attempting to extend standard algorithms used in structural dynamics to the area of multibody dynamics. The stiff nature of the differential-algebraic equations encountered in multibody systems make the stabilizing feature of energy conserving algorithms almost a necessity in long duration simulations. In general the energy conserving methods in structural dynamics require only special treatment of the internal force terms, those which are dependent on the material model under consideration. In the DAE case however the additional term in the equation

of motion, which comes from imposition of the constraints, must also be carefully constructed as to not destroy the energy conserving nature of the algorithm. Recently it has been shown by Betsch [17] that for constraint equations that are quadratic or lower in the degrees of freedom maintaining the energy conserving property of an algorithm requires only careful evaluation of the additional term, no additional reformulation of the existing algorithm. This method of energy conservation for DAEs, detailed further in the following chapters, is the foundation of the research herein.

Chapter 2

Structural Dynamics

The intention of this chapter is to provide readers an overview of previous work regarding the extension of the GSSSS family of algorithms to nonlinear systems of ordinary differential equations. This background is necessary to understand their further extension to the solution of differential-algebraic equations, the core motivation of this research. First an overview of the GSSSS algorithms is given, followed by the normalized weighted residual method for development of nonlinear algorithms, and finally the importance of consistency of the equation of motion time level.

2.1 The Framework of the GSSSS Family of Algorithms Encompassing LMS Methods

Recently, a novel concept that provides new avenues to design time integration operators for linear dynamic problems via a unified framework, which unifies all of the existing algorithms known to the author and provides new designs of time operators encompassing the class of LMS methods involving a single solve within a given time step, and second order time accuracy was introduced in a series of papers by Zhou et al. [18–22]; a notion called algorithms by design. By choosing

certain key algorithmic parameters called the algorithmic DNA markers, the unified framework under the umbrella of the family of generalized single solve single step (GSSSS) time operators was uniquely designed. By simply and properly selecting only the two principal roots and the spurious root, the underlying design enabled one to not only recover all of the existing time operators in the literature in the sense of LMS methods including non-dissipative algorithms such two versions of the Midpoint rule: endpoint acceleration and midpoint acceleration [23], and the Newmark average acceleration method [24], but also additionally provided new avenues towards new algorithmic designs with optimal features. This, is in the sense of the least amount of algorithmic dissipation, dispersion, and overshoot behavior. In general, for the single field form of representation involving a single solve within each time step, in the sense of LMS methods, there exist three roots participating in the algorithm, namely, the two principal roots $\rho_{1\infty}$, $\rho_{2\infty}$ and the spurious root $\rho_{3\infty}$. It has been shown in [20, 22] that the second-order time accurate, unconditionally stable LMS framework is basically comprised of two distinct algorithmic structures, termed as constrained U (displacement overshooting aspect, also referred to as U0) and constrained V (velocity overshooting aspect, also referred to as V0) family of algorithms for linear structural dynamic systems. Most of the developments to-date for both numerically non-dissipative and dissipative algorithms pertain to the constrained U-family of algorithms within the LMS class of methods for finite element computations.

Most of the traditionally developed time integration operators appearing in the literature to-date for linear dynamic problems, belong only to the U0 family such as the non-dissipative Newmark method [24] and midpoint rule with endpoint acceleration, while the controllable dissipative algorithms such as the WBZ [25] and HHT-α [26], and the like exist(see various references contained in [20, 22]. This means that the relatively unknown V0 family has, for the most part, not been the focus of significant research and almost certainly has not been considered in the case of index-3 DAEs.

2.1.1 The U0 and V0 algorithms

The U0 family of algorithms contains all the algorithms with zero-order displacement overshooting behavior, and the V0 family of algorithms contains all the algorithms with zero-order velocity overshooting behavior. These two families of algorithms have been derived and described in [22]. However, for the purpose of easy reference in this chapter, the U0 family and the V0 family are highlighted next.

Consider the semi-discretized system of equations of linear structural dynamic problems by space discretization of the single field form as:

$$\begin{aligned} \mathbf{M}\ddot{\mathbf{u}}(t) + \mathbf{C}\dot{\mathbf{u}}(t) + \mathbf{K}\mathbf{u}(t) &= \mathbf{f}(t) \\ \mathbf{u}(0) = \mathbf{u}_0 \quad , \quad \dot{\mathbf{u}}(0) &= \dot{\mathbf{u}}_0 \end{aligned} \tag{2.1}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{C}$ is the damping matrix, and $\mathbf{K}$ is the stiffness matrix. The U0 family of algorithms is given as follows.

Algorithm 1 (U0 Family of Algorithms)
Given $\mathbf{u}_n$, $\dot{\mathbf{u}}_n$, *and* $\ddot{\mathbf{u}}_n$, *find* $\mathbf{u}_{n+1}$, $\dot{\mathbf{u}}_{n+1}$, *and* $\ddot{\mathbf{u}}_{n+1}$ *from*

$$\begin{aligned} & \left(\Lambda_6 W_1 \mathbf{M} + \Lambda_5 W_2 \mathbf{C}\Delta t + \Lambda_3 W_3 \mathbf{K}\Delta t^2\right)\Delta\mathbf{a} \\ = \; & -\mathbf{M}\ddot{\mathbf{u}}_n - \mathbf{C}\left(\dot{\mathbf{u}}_n + \Lambda_4 W_1 \ddot{\mathbf{u}}_n \Delta t\right) \\ & -\mathbf{K}\left(\mathbf{u}_n + \Lambda_1 W_1 \dot{\mathbf{u}}_n \Delta t + \Lambda_2 W_2 \ddot{\mathbf{u}}_n \Delta t^2\right) \\ & +(1 - W_1)\mathbf{f}_n + W_1 \mathbf{f}_{n+1} \end{aligned}$$

$$\begin{aligned} \mathbf{u}_{n+1} &= \mathbf{u}_n + \lambda_1 \dot{\mathbf{u}}_n \Delta t + \lambda_2 \ddot{\mathbf{u}}_n \Delta t^2 + \lambda_3 \Delta\mathbf{a}\Delta t^2 \\ \dot{\mathbf{u}}_{n+1} &= \dot{\mathbf{u}}_n + \lambda_4 \ddot{\mathbf{u}}_n \Delta t + \lambda_5 \Delta\mathbf{a}\Delta t \\ \ddot{\mathbf{u}}_{n+1} &= \ddot{\mathbf{u}}_n + \Delta\mathbf{a} \end{aligned}$$

where

$$
\begin{aligned}
W_1\Lambda_1 &= \frac{1}{1+\rho_{3\infty}} \quad , \quad \lambda_1 = 1 \\
W_2\Lambda_2 &= \frac{1}{2(1+\rho_{3\infty})} \quad , \quad \lambda_2 = \frac{1}{2} \\
W_3\Lambda_3 &= \frac{1}{(1+\rho_{1\infty})(1+\rho_{2\infty})(1+\rho_{3\infty})} \quad , \quad \lambda_3 = \frac{1}{(1+\rho_{1\infty})(1+\rho_{2\infty})} \\
W_1\Lambda_4 &= \frac{1}{1+\rho_{3\infty}} \quad , \quad \lambda_4 = 1 \\
W_2\Lambda_5 &= \frac{3+\rho_{1\infty}+\rho_{2\infty}-\rho_{1\infty}\rho_{2\infty}}{2(1+\rho_{1\infty})(1+\rho_{2\infty})(1+\rho_{3\infty})} \quad , \quad \lambda_5 = \frac{3+\rho_{1\infty}+\rho_{2\infty}-\rho_{1\infty}\rho_{2\infty}}{2(1+\rho_{1\infty})(1+\rho_{2\infty})} \\
W_1\Lambda_6 &= \frac{2+\rho_{1\infty}+\rho_{2\infty}+\rho_{3\infty}-\rho_{1\infty}\rho_{2\infty}\rho_{3\infty}}{(1+\rho_{1\infty})(1+\rho_{2\infty})(1+\rho_{3\infty})}
\end{aligned}
$$

The weighted time field is suggested to be,

$$
W = 1 - \frac{15(1-2\rho_{3\infty})}{1-4\rho_{3\infty}}\frac{\tau}{\Delta t} + \frac{15(3-4\rho_{3\infty})}{1-4\rho_{3\infty}}\left(\frac{\tau}{\Delta t}\right)^2 - \frac{35(1-\rho_{3\infty})}{1-4\rho_{3\infty}}\left(\frac{\tau}{\Delta t}\right)^3 ; \ \tau \in [0, \Delta t]
$$

and

$$
W_i = \frac{\sum_{j=0}^{3} \frac{w_j}{1+i+j}}{\sum_{j=0}^{3} \frac{w_j}{1+j}} \quad ; \quad i = 1, 2, 3
$$

$\rho_{1\infty}$, $\rho_{2\infty}$, and $\rho_{3\infty}$ are the first principal root, the second principal root, and the spurious root at the high-frequency limit, respectively, and they satisfy the following relation:

$$
0 \leq \rho_{3\infty} \leq \rho_{1\infty} \leq \rho_{2\infty} \leq 1
$$

Additionally, the V0 family of algorithms is given as follows.

Algorithm 2 (V0 Family of Algorithms)

Given $\mathbf{u}_n$, $\dot{\mathbf{u}}_n$, and $\ddot{\mathbf{u}}_n$, find $\mathbf{u}_{n+1}$, $\dot{\mathbf{u}}_{n+1}$, and $\ddot{\mathbf{u}}_{n+1}$ from

$$
\begin{aligned}
& \left(\Lambda_6 W_1 \mathbf{M} + \Lambda_5 W_2 \mathbf{C}\Delta t + \Lambda_3 W_3 \mathbf{K}\Delta t^2\right)\Delta\mathbf{a} \\
= \ & -\mathbf{M}\ddot{\mathbf{u}}_n - \mathbf{C}\left(\dot{\mathbf{u}}_n + \Lambda_4 W_1 \ddot{\mathbf{u}}_n \Delta t\right) \\
& -\mathbf{K}\left(\mathbf{u}_n + \Lambda_1 W_1 \dot{\mathbf{u}}_n \Delta t + \Lambda_2 W_2 \ddot{\mathbf{u}}_n \Delta t^2\right) \\
& +(1-W_1)\mathbf{f}_n + W_1 \mathbf{f}_{n+1}
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{u}_{n+1} &= \mathbf{u}_n + \lambda_1 \dot{\mathbf{u}}_n \Delta t + \lambda_2 \ddot{\mathbf{u}}_n \Delta t^2 + \lambda_3 \Delta \mathbf{a} \Delta t^2 \\
\dot{\mathbf{u}}_{n+1} &= \dot{\mathbf{u}}_n + \lambda_4 \ddot{\mathbf{u}}_n \Delta t + \lambda_5 \Delta \mathbf{a} \Delta t \\
\ddot{\mathbf{u}}_{n+1} &= \ddot{\mathbf{u}}_n + \Delta \mathbf{a}
\end{aligned}
$$

where

$$
\begin{aligned}
W_1\Lambda_1 &= \frac{3+\rho_{1\infty}+\rho_{2\infty}-\rho_{1\infty}\rho_{2\infty}}{2(1+\rho_{1\infty})(1+\rho_{2\infty})} \quad , \quad \lambda_1 = 1 \\
W_2\Lambda_2 &= \frac{1}{(1+\rho_{1\infty})(1+\rho_{2\infty})} \quad , \quad \lambda_2 = \frac{1}{2} \\
W_3\Lambda_3 &= \frac{1}{(1+\rho_{1\infty})(1+\rho_{2\infty})(1+\rho_{3\infty})} \quad , \quad \lambda_3 = \frac{1}{2(1+\rho_{3\infty})} \\
W_1\Lambda_4 &= \frac{3+\rho_{1\infty}+\rho_{2\infty}-\rho_{1\infty}\rho_{2\infty}}{2(1+\rho_{1\infty})(1+\rho_{2\infty})} \quad , \quad \lambda_4 = 1 \\
W_2\Lambda_5 &= \frac{2}{(1+\rho_{1\infty})(1+\rho_{2\infty})(1+\rho_{3\infty})} \quad , \quad \lambda_5 = \frac{1}{1+\rho_{3\infty}} \\
W_1\Lambda_6 &= \frac{2+\rho_{1\infty}+\rho_{2\infty}+\rho_{3\infty}-\rho_{1\infty}\rho_{2\infty}\rho_{3\infty}}{(1+\rho_{1\infty})(1+\rho_{2\infty})(1+\rho_{3\infty})}
\end{aligned}
$$

The weighted time field is suggested to be,

$$
\begin{aligned}
W =& 1 - \frac{30(3-4\rho_{1\infty}-4\rho_{2\infty}+6\rho_{1\infty}\rho_{2\infty})}{9-11\rho_{1\infty}-11\rho_{2\infty}+19\rho_{1\infty}\rho_{2\infty}} \frac{\tau}{\Delta t} \\
&+ \frac{15(25-37\rho_{1\infty}-37\rho_{2\infty}+53\rho_{1\infty}\rho_{2\infty})}{2(9-11\rho_{1\infty}-11\rho_{2\infty}+19\rho_{1\infty}\rho_{2\infty})} \left(\frac{\tau}{\Delta t}\right)^2 \\
&- \frac{35(3-5\rho_{1\infty}-5\rho_{2\infty}+7\rho_{1\infty}\rho_{2\infty})}{9-11\rho_{1\infty}-11\rho_{2\infty}+19\rho_{1\infty}\rho_{2\infty}} \left(\frac{\tau}{\Delta t}\right)^3 ; \\
& \tau \in [0, \Delta t]
\end{aligned}
$$

and

$$
W_i = \frac{\sum_{j=0}^{3} \frac{w_j}{1+i+j}}{\sum_{j=0}^{3} \frac{w_j}{1+j}} \quad ; \quad i = 1, 2, 3
$$

$\rho_{1\infty}$, $\rho_{2\infty}$, and $\rho_{3\infty}$ are the first principal root, the second principal root, and the spurious root at the high-frequency limit, respectively, and they satisfy the following relation:

$$
0 \leq \rho_{3\infty} \leq \rho_{1\infty} \leq \rho_{2\infty} \leq 1
$$

2.1.2 Usage of the GSSSS Algorithms

It is the above two families of linear algorithms which are the basis of the research within. Under the framework of the GSSSS family of algorithms, all of the currently existing and more recent new developments of optimal algorithms that are second order time accurate encompassing LMS methods can be constructed by first selecting the family (either the so-called U0 family or V0 family), and then the three parameters $\rho_{1\infty}$, $\rho_{2\infty}$, and $\rho_{3\infty}$ specify the particular algorithm that is selected. Simply for illustration, Table 2.1 highlights some of the more commonly known algorithms, and how to implement them under the GSSSS framework.

Table 2.1: Commonly known algorithms

Algorithm Family($\rho_{1\infty}$,$\rho_{2\infty}$,$\rho_{3\infty}$)	Common name	Conditions $0 \leq \rho_{i\infty} \leq 1$
U0(1,1,0)	Newmark	-
U0/V0(1,1,1)	Midpoint Rule endpoint accel.	-
U0($\rho_{1\infty}$,$\rho_{2\infty}$,$\rho_{3\infty}$)	Three-parameter optimal scheme (Generalized-α)	$\rho_{1\infty} = \rho_{2\infty} = \rho_{3\infty} = \rho_\infty$
U0($\rho_{1\infty}$,$\rho_{2\infty}$,0)	WBZ	$\rho_{1\infty} = \rho_{2\infty} = \rho_\infty$
U0($\rho_{1\infty}$,$\rho_{2\infty}$,$\rho_{3\infty}$)	HHT-α	$\rho_{1\infty} = \rho_{2\infty}, \rho_{3\infty} = \frac{1-\rho_{1\infty}\rho_{2\infty}}{\rho_{1\infty}+\rho_{2\infty}+2\rho_{1\infty}\rho_{2\infty}}$
U0/V0($\rho_{1\infty}$,1,$\rho_{3\infty}$)	U0-V0 Optimal	$\rho_{1\infty} = \rho_{3\infty} = \rho_\infty$
V0(1,1,0)	Midpoint Rule midpoint accel.	-

2.1.3 Equation of Motion Time level

In previous work [27] the importance of the equation of motion (EOM) time level was demonstrated. The time level at which the equation of motion needs to be computed is a very fundamental and important concept to understand, especially in the case of providing extensions of linear dynamic algorithms to nonlinear dynamic problems. Improper calculation of time dependent variables which are

included in the equation of motion can lead to decreased accuracy of the algorithm, and also a failure to conserve key quantities in algorithm designs which seek to establish energy and momentum conservation. To understand its importance and significance, we must first define the meaning and implication regarding the time level for the equation of motion. Fundamentally, computational structural dynamics problems first comprise of the governing strong form of the partial differential equations of motion. The semi-discrete weak form of the equation of motion is obtained through spatial discretization. This yields a system of ordinary differential equations which can be solved computationally. The ordinary differential equations arising from the semi-discretized problem are then discretized in time leading to various forms of representation for the underlying algorithm. In the linear dynamic cases, this fully discretized equation of motion has the form:

$$M\tilde{a}(t) + C\tilde{v}(t) + K\tilde{u}(t) = \tilde{F}(t) \tag{2.2}$$

This equation must be valid at any time t. The goal of the framework of the GSSSS family of algorithms is to take any spatially discretized system of equations, and therein provide a means to march forward in time. All that is required are the set of given initial conditions $u(t_0)$, $v(t_0)$, and it is assumed the matrices M, C, K, and vector F are known from the spatially discretized mesh. Given these initial values, every algorithm within the entire framework of the GSSSS family of algorithms, then solves the equation of motion at some key time point, namely, $\tilde{t}$, where $t_n \leq \tilde{t} \leq t_{n+1.5}$, and subsequently uses an asymptotic series type expansion to approximate the values of u and v at time t_{n+1}. Specifically, it is this key time point, namely, $\tilde{t}$ that is here referred to as the equation of motion time level; it is defined as the discrete time point precisely at which the equation of motion must be satisfied.

It was shown specifically that the framework of the GSSSS family of algorithms satisfies the equation of motion at the time t_{n+W_1} for time dependent variables v, u, and F. The value of a appeared to be inconsistently calculated not at time t_{n+W_1} but instead at time $t_{n+W_1\Lambda 6}$. In [28] this seemingly inconsistent time

level evaluation was explained, and it was shown that the acceleration is in fact also calculated at time t_{n+W_1}. This specific time level must be satisfied for *any* algorithm that is designed from the second order accurate framework of the GSSSS family of algorithms. This concept was essential in this research, particularly in the extension to the nonlinear realm, to derive consistent second order accurate algorithms for use in solution of the differential-algebraic equations encountered in the area of interest: multibody dynamics.

2.1.4 Extensions to Nonlinear Equations

Having described that both the U0 and V0 families of algorithms satisfy the equation of motion at a consistent time level t_{n+W_1}, the importance of this time level for proper extension to nonlinear equations must be understood. For the linear dynamic cases, if the proper time level at which the equation of motion needs to be satisfied for a given algorithm under consideration is not taken into account, it will lead to degradation in the order of accuracy of the accelerations in the framework of the GSSSS family of algorithms. Nonlinear dynamic applications require further attention be paid to this time level. It was shown that algorithmic designs and developments in linear dynamic situations are very valuable, and indeed these parent linear dynamic algorithms form the basis for extensions to the case of non-linear dynamic problems. How to extend such developments to nonlinear dynamics applications needs careful attention, and literature has not been very careful or consistent in drawing the subtle, but important distinctions in enacting these extensions. In this regard, for nonlinear dynamic applications, a variety of techniques were carefully described and analyzed in [28]- [29]. The three main methods were the classical time weighted residual, symplectic-momentum based, and energy-momentum based algorithms. Of these, the energy-momentum based algorithms are the focus of this research due to their reputation for providing stable solutions for multibody simulation.

Recall that for the general case of nonlinear dynamic problems the equation

of motion takes the following form.

$$M\tilde{a}(t) + C\tilde{v}(t) + \tilde{p}(t) = \tilde{F}(t)$$

For the energy-momentum based algorithmic designs of a geometrically nonlinear truss model [14] the approximations of the strain and displacement are independent, and these approximations need to be constructed carefully. To be consistent with displacements, velocities, and accelerations (shown to be at time level t_{n+W_1}), the strain would need to be also approximated at the same time level. It was described that in the general case of the GSSSS framework of algorithms, in particular for the controllable dissipative algorithms, there did not exist a viable justification on how to correctly implement the approximation for strain. The equation of motion time level concept provided this basis and was able to provide a means to enable extensions to nonlinear dynamics applications via the normalized weighted residual technique.

2.1.5 Energy-Momentum Methodology

The energy-momentum methodology, described in detail in [30], involves a hybrid procedure where the strain is approximated independently from the displacement. For the U0 family of algorithms it defines the strain to be a linear approximation between t_n and t_{n+1}. The approximations for $\ddot{u}$ and u come directly from the parent linear dynamic algorithms. The importance of understanding the EOM time level now is evident from the strain term. Without knowledge of the EOM time level, simply stating that we wish to approximate strain by a trapezoidal rule representation between t_n and t_{n+1} provides no basis for *where* to choose the time point between t_n and t_{n+1}. Considering the point collocation at time level t_{n+W_1}, we now have a justification for the time point at which to approximate the strain. It has been shown extensively in [27] that correctly evaluating the strain at time level t_{n+W_1} results in algorithms which are second order accurate in all variables, whereas any shift from this correct time level destroys the accuracy of the algorithms as well as the energy conserving properties.

The key outcome of the previous research on energy and momentum algorithms is the absolute necessity of time level collocation of any additional terms. Here, that was manifest in the development of a method where the strain was independently approximated from the other primary variables. Such an approximation has no basis without this time level concept and any algorithm which deviates from this time level has been shown to suffer from loss of accuracy and loss of conservation properties. In the following chapter, when this method is extended to include constrained dynamics, this concept is essential. The generic algorithm for the energy-momentum based method is shown in below. Notice that the strain term, a displacement level in time term, is approximated in exactly the same way as the unknown displacement: at time t_{n+W_1}.

At the beginning of time step, predict the state vectors

$$\mathbf{u}^k_{n+1} = \chi^{(1)}_{Pu}\ \mathbf{u}_n + \chi^{(2)}_{Pu}\ \dot{\mathbf{u}}_n + \chi^{(3)}_{Pu}\ \ddot{\mathbf{u}}_n \tag{2.3}$$

$$\widetilde{\mathbf{u}}^k_{n+1} = \chi^{(1)}_{P\tilde{u}}\ \mathbf{u}_n + \chi^{(2)}_{P\tilde{u}}\ \dot{\mathbf{u}}_n + \chi^{(3)}_{P\tilde{u}}\ \ddot{\mathbf{u}}_n \tag{2.4}$$

$$\widetilde{\dot{\mathbf{u}}}^k_{n+1} = \chi^{(1)}_{P\tilde{v}}\ \mathbf{u}_n + \chi^{(2)}_{P\tilde{v}}\ \dot{\mathbf{u}}_n + \chi^{(3)}_{P\tilde{v}}\ \ddot{\mathbf{u}}_n \tag{2.5}$$

$$\widetilde{\ddot{\mathbf{u}}}^k_{n+1} = \chi^{(1)}_{P\tilde{a}}\ \mathbf{u}_n + \chi^{(2)}_{P\tilde{a}}\ \dot{\mathbf{u}}_n + \chi^{(3)}_{P\tilde{a}}\ \ddot{\mathbf{u}}_n \tag{2.6}$$

Start nonlinear iteration. Solving for $\Delta\delta^{k+1}_{n+1}$ from

$$\begin{aligned}&[\chi_{Ca}\mathbf{M} + \chi_{Cv}\mathbf{C} + \chi_{Cd}\,{}^t\mathbf{K}^k_{n+1}]\Delta\delta^{k+1}_{n+1} = \\ &-(\mathbf{M}\widetilde{\ddot{\mathbf{u}}}^k_{n+1} + \mathbf{C}\widetilde{\dot{\mathbf{u}}}^k_{n+1} + \mathbf{K}_1\widetilde{\mathbf{u}}^k_{n+1} + \mathbf{K}_2\widetilde{\epsilon}_{n+1} + \mathbf{K}_3\widetilde{\mathbf{u}}^k_{n+1}\widetilde{\epsilon}_{n+1} - \widetilde{\mathbf{F}}^k_{n+1})\end{aligned} \tag{2.7}$$

$${}^t\mathbf{K}^k_{n+1} = \mathbf{K}_1 + \mathbf{K}_2(\mathbf{b} + \frac{4}{l_0^2}\mathbf{A}\mathbf{u}^k_{n+1}) + \mathbf{K}_3[\widetilde{\mathbf{u}}^k_{n+1}(\mathbf{b} + \frac{4}{l_0^2}\mathbf{A}\mathbf{u}^k_{n+1}) + \widetilde{\epsilon}_{n+1}] \tag{2.8}$$

where the element based EOM time level strain $\widetilde{\epsilon}_{n+1}$ is evaluated as:

$$\widetilde{\epsilon}_{n+1} = \chi^{(1)}_{P\tilde{u}}\ \epsilon_n + \chi^{(2)}_{P\tilde{u}}\ \dot{\epsilon}_n + \chi^{(3)}_{P\tilde{u}}\ \ddot{\epsilon}_n + \chi_{C\tilde{u}}\left(\epsilon(\mathbf{u}^k_{n+1}) - \epsilon(\mathbf{u}_n)\right) \tag{2.9}$$

Then correct the primary variables as follows:

$$\mathbf{u}_{n+1}^{k+1} = \mathbf{u}_{n+1}^{k} + \chi_{Cu}\Delta\boldsymbol{\delta}_{n+1}^{k+1} \tag{2.10}$$

$$\widetilde{\mathbf{u}}_{n+1}^{k+1} = \widetilde{\mathbf{u}}_{n+1}^{k} + \chi_{C\tilde{u}}\Delta\boldsymbol{\delta}_{n+1}^{k+1} \tag{2.11}$$

$$\widetilde{\dot{\mathbf{u}}}_{n+1}^{k+1} = \widetilde{\dot{\mathbf{u}}}_{n+1}^{k} + \chi_{C\tilde{v}}\Delta\boldsymbol{\delta}_{n+1}^{k+1} \tag{2.12}$$

$$\widetilde{\ddot{\mathbf{u}}}_{n+1}^{k+1} = \widetilde{\ddot{\mathbf{u}}}_{n+1}^{k} + \chi_{C\tilde{a}}\Delta\boldsymbol{\delta}_{n+1}^{k+1} \tag{2.13}$$

until the solution converges

$$\left|\delta_{n+1}^{k+1} - \delta_{n+1}^{k}\right| < tolerance \tag{2.14}$$

Once the solution converges, update the primary variables at the end of the time step as follows.

$$\ddot{\mathbf{u}}_{n+1} = \ddot{\mathbf{u}}_{n} + (\tilde{\ddot{\mathbf{u}}}_{n+1}^{k+1} - \ddot{\mathbf{u}}_{n})/(\Lambda_6 W_1) \tag{2.15}$$

$$\dot{\mathbf{u}}_{n+1} = \dot{\mathbf{u}}_{n} + \lambda_4\ddot{\mathbf{u}}_{n}\Delta t + \lambda_5(\ddot{\mathbf{u}}_{n+1} - \ddot{\mathbf{u}}_{n})\Delta t \tag{2.16}$$

$$\mathbf{u}_{n+1} = \mathbf{u}_{n} + \lambda_1\dot{\mathbf{u}}_{n}\Delta t + \lambda_2\ddot{\mathbf{u}}_{n}\Delta t^2 + \lambda_3(\ddot{\mathbf{u}}_{n+1} - \ddot{\mathbf{u}}_{n})\Delta t^2 \tag{2.17}$$

The predictor-corrector coefficients χ above in the corresponding a, v, and d-form are listed in Table 2.2.

It is useful to note that this methodology describes a generic algorithmic framework which includes energy and momentum conserving algorithms, but also includes algorithms with controllable numerical dissipation. As well, it produces and entire family of algorithms which conserve energy and momentum. Any algorithm of the V0 family with the values of $\rho_{1\infty} = \rho_{2\infty} = 1$ will result in an exact energy and momentum preserving algorithm using the above framework. The fundamental reason for this is that any algorithm within the V0 family with $\rho_{1\infty} = \rho_{2\infty} = 1$ satisfies the equation of motion exactly at $t_{n+\frac{1}{2}}$. Gonzalez [16] provides a theoretical basis for this concept, which essentially says that for any linear material model evaluation of the equation of motion at $t_{n+\frac{1}{2}}$ will result in energy conservation. The independent approximation of strain described above is required to obtain energy conservation in this case due to the necessity of a

nonlinear strain definition resulting from very large rotations. A linear strain definition would allow simple evaluation of the internal force term directly from the midpoint displacement. Any algorithm not resulting in consistent evaluation of all terms at the midpoint will no longer display the conservative properties. It is worth noting that for nonlinear materials (those which are not simply quadratic in their potential energy) simply evaluating internal force at the midpoint is no longer sufficient to obtain energy conservation. Techniques for dealing with such nonlinearities are given in [7] and [12].

Table 2.2: Predictor multi-corrector coefficients for the incremental a-, v- and d-form representations

	a-form	v-form	d-form
$\chi_{Pu}^{(1)}$	1	1	1
$\chi_{Pu}^{(2)}$	$\lambda_1\Delta t$	$\lambda_1\Delta t$	0
$\chi_{Pu}^{(3)}$	$\lambda_2\Delta t^2$	$(\lambda_2 - \frac{\lambda_3\lambda_4}{\lambda_5})\Delta t^2$	0
$\chi_{P\tilde{u}}^{(1)}$	1	1	1
$\chi_{P\tilde{u}}^{(2)}$	$\Lambda_1 W_1\Delta t$	$\Lambda_1 W_1\Delta t$	$(\Lambda_1 W_1 - \frac{\Lambda_3 W_3\lambda_1}{\lambda_3})\Delta t$
$\chi_{P\tilde{u}}^{(3)}$	$\Lambda_2 W_2\Delta t^2$	$(\Lambda_2 W_2 - \frac{\Lambda_3 W_3\lambda_4}{\lambda_5})\Delta t^2$	$(\Lambda_2 W_2 - \frac{\Lambda_3 W_3\lambda_2}{\lambda_3})\Delta t^2$
$\chi_{P\tilde{v}}^{(1)}$	0	0	0
$\chi_{P\tilde{v}}^{(2)}$	1	1	$1 - \frac{\Lambda_5 W_2\lambda_1}{\lambda_3}$
$\chi_{P\tilde{v}}^{(3)}$	$\Lambda_4 W_1\Delta t$	$(\Lambda_4 W_1 - \frac{\Lambda_5 W_2\lambda_4}{\lambda_5})\Delta t$	$(\Lambda_4 W_1 - \frac{\Lambda_5 W_2\lambda_2}{\lambda_3})\Delta t$
$\chi_{P\tilde{a}}^{(1)}$	0	0	0
$\chi_{P\tilde{a}}^{(2)}$	0	0	$-\frac{\Lambda_6 W_1\lambda_1}{\lambda_3\Delta t}$
$\chi_{P\tilde{a}}^{(3)}$	1	$1 - \frac{\Lambda_6 W_1\lambda_4}{\lambda_5}$	$1 - \frac{\Lambda_6 W_1\lambda_2}{\lambda_3}$
χ_{Cu}	$\lambda_3\Delta t^2$	$\frac{\lambda_3}{\lambda_5}\Delta t$	1
$\chi_{C\tilde{u}}$	$\Lambda_3 W_3\Delta t^2$	$\frac{\Lambda_3 W_3}{\lambda_5}\Delta t$	$\frac{\Lambda_3 W_3}{\lambda_3}$
$\chi_{C\tilde{v}}$	$\Lambda_5 W_2\Delta t$	$\frac{\Lambda_5 W_2}{\lambda_5}$	$\frac{\Lambda_5 W_2}{\lambda_3\Delta t}$
$\chi_{C\tilde{a}}$	$\Lambda_6 W_1$	$\frac{\Lambda_6 W_1}{\lambda_5\Delta t}$	$\frac{\Lambda_6 W_1}{\lambda_3\Delta t^2}$

Chapter 3

Constrained Dynamics

3.1 Motivation

The fundamental difference between the elastodynamics of the previous chapter and constrained dynamics is the presence of algebraic equations within the mathematical model. In constrained dynamics, the constraints are simply additional algebraic equations coupled to the unconstrained equations of motion most commonly via Lagrange multipliers. Though a variety of other techniques for coupling the algebraic constraint equations to the original differential equations of motion exist within the literature, the Lagrange multiplier method is the most commonly used and is the focus of this research. This coupling of the constraint equations to the ordinary differential equations of elastodynamics transforms the system to a set of differential-algebraic equations (DAEs). The study of numerical solutions to differential-algebraic equations is a relatively young area of research, with most of the major developments occurring within the last 25 years. Research in the area has come from a variety of fields where DAEs are encountered frequently: constrained dynamics, molecular dynamics, electrical engineering, and chemistry. Most recently, simulation of constrained mechanical systems such as wind turbines and helicopter rotors has become an area of significant research efforts.

3.2 Difficulties in DAEs

Numerical solutions to differential algebraic equations pose additional difficulties not found in the solution of their ODE counterparts, but also inherit many of the same properties and problems. In order to discuss these additional difficulties and the current state of overcoming them one needs first to understand how DAEs are classified. The most important property of a DAE is its so-called "index". Petzold [31] defines the index of the DAE as:

> The minimum number of times that all or part of [the DAE] must be differentiated with respect to t in order to determine y' as a continuous function of y, t, is the index of the DAE.

where $y(t)$ is the solution to some general DAE $F(t, y, y') = 0$. That is, the index is the number of times you need to differentiate the DAE in order to recover a system which can be written as a standard ODE.

Within the area of multibody dynamics (MBD) the equations of motion coupled to the constraint equations naturally give rise to a set of index 3 DAEs, due to the constraint equations being constraints on the displacement of the system (as opposed to velocity or acceleration constraints). This is the most common form of constraint and includes revolute joints, sliding contact, rolling contact, and many other common joints between two structural bodies. Though these index 3 DAEs are quite natural to MBD systems, solving them directly in this form is uncommon in the literature. The majority of the codes used in solving MBD problems reformulate the natural index 3 equations into either index 2 or index 1 formulations. It is commonly noted that the higher the index of the DAE, the more difficult it is to obtain a stable numerical solution. Many authors cite stiffness, instability, and order reduction as reasons to avoid solving the index 3 problem directly. Though this seems to be a generally accepted idea, it is rare to find an explanation of the source of these issues. That said, multibody systems of any practical interest almost certainly involve large rotation or nonlinear

constraints (resulting in nonlinear system equations), making any sort of analytic proof nearly impossible.

Examples of simulations run directly on index 3 DAEs which "blow up" preface almost every paper introducing a new index 2 or index 1 algorithm. Even the most fundamental of MBD problems, dynamic response of a simple pendulum, when run using existing methods will fail. That is not to say there are not specific index 3 DAE solvers that can solve the problem, but these solvers either use generalized coordinates (not Cartesian coordinates as required by most standard finite element codes) or use controllable numerical dissipation which artificially removes energy from the system. More often, this unbounded oscillation is the basis for developing algorithms in index 2 or 1 which are able to avoid the problem. Borri [2] demonstrates this failure for the simple pendulum by showing that the numerical solution to the index 3 problem (without dissipation), though able to produce smooth and stable displacements and velocities, will develop oscillations in the accelerations (and therefore Lagrange multipliers) as seen in Fig. 3.1. Also, it is noted that the order of accuracy of the Lagrange multipliers will suffer a reduction with respect to the displacements and velocities of the system.

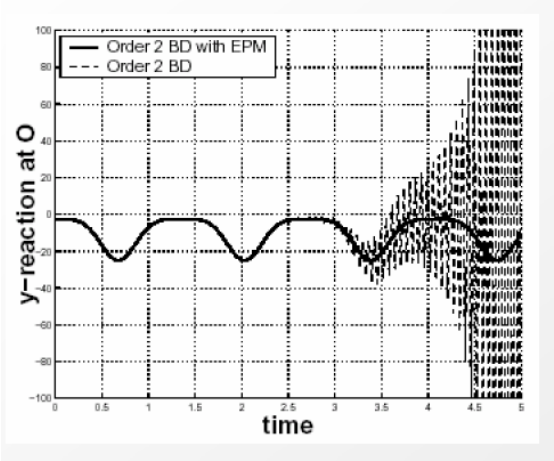


Figure 3.1: Single Pendulum: Constraint Force Oscillations from [2]

Another example, this time of a double pendulum, is discussed at length by Geradin in [3]. Using the (non-dissipative) Newmark method it is again shown that the accelerations and Lagrange multipliers exhibit unbounded oscillatory behavior, see Fig. 3.2. These unbounded oscillations eventually lead to convergence problems in the nonlinear iterations and the solution is unable to proceed. In attempt to remedy these unstable oscillation it is later shown that using the HHT-α method removes the unwanted (and nonphysical) oscillations, but at the price of dissipating energy from what should be a conservative system.

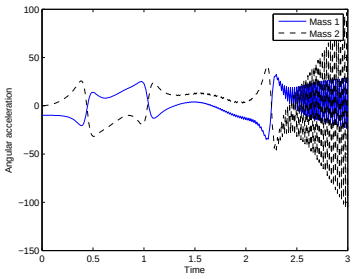


Figure 3.2: Newmark Method: Double Pendulum Acceleration blow-up from [3]

Though the focus of the current research is solely on solving index 3 DAEs directly, it is worth mentioning some of the methods used to reduce the system to index 2 or 1 and the drawbacks of doing so.

The most basic way to lower the index of a DAE is to simply differentiate the constraint equation with respect to time. Assuming we start with a constraint equation on the displacements, one such differentiation will result in an index 2 DAE where the constraint equation now involves the time derivative of the displacement, the velocity. One more differentiation will result in an index 1 DAE where the constraint equation involves the accelerations of the system. Upon implementation of standard BDF or RK time integration algorithms these index 2 and index 1 systems no longer exhibit the unbounded oscillations which occur in the index 3 system. Unfortunately, while correcting one issue index reduction

introduces a new issue: constraint drift. This problem is, among a variety of other places, described by Hairer in [32]. The fundamental problem is that when the constraint equation occurs at either the velocity or acceleration (index 2 or index 1 respectively) the original position level constraint "'drifts"' away from its true solution. That is, the original index 3 system which defines the mathematical model for the physical system under consideration is no longer being satisfied to the nonlinear tolerance.

To deal with the problem of constraint drift a wide variety of techniques have been developed. Arguably the most famous and widely known is termed Baumgarte stabilization [33]. Baumgarte proposed a method which combined the position, velocity, and acceleration constraint into a single equation which was originally used in control theory. This method includes two parameters which are used to control the combined constraint equation. Though optimal values of these parameters which guarantee solution stability often do exist, it is required that the user is somehow able to determine them. With no robust mathematical method to determine these parameters the user is likely to simply use a guess and check method to find what works empirically, often not an efficient process.

Another common way of dealing with constraint drift are the so-called "projection methods". First made popular by Lubich [34], these methods work by first solving the reduced index system and then projecting that solution back onto the solution manifold of the original index 3 system, from which it has drifted away. Extensions of the original method abound in the literature, of the more famous is the index 1 DAE code DASSL [35] which is still popular today. These projection methods are again, in general, able to overcome the instability seen in the index 3 solution but at the cost of a) reformulation by index reduction, b) additional computational cost of the projections.

The final method commonly used to overcome drift of the constraint equation within a reduced index formulation is again to take the derivative of the constraint equations with respect to time, but instead of using only the first or second (velocity or acceleration level) derivative equation, the original position

constraint is used as well. That is, the original equations of motion are coupled with not only the position level constraint, but also with one or possibly both of the velocity and acceleration level constraints. In this case, of course, there are now more equations than unknowns which is referred to as an overdetermined system. This formulation has garnered significant support after being introduced by Gear in [36]. Recently both Lunk [37] and Negrut [38] have used this method in conjunction with popular time integration algorithms from structural dynamics successfully.

All of these methods were developed to overcome the initial problem of instability in the natural index 3 formulation. Though capable of removing the problem of acceleration instability, they introduce error in the position level (index 3) constraint which then requires additional effort to remove. That is to say, in the opinion of the author, if there were an algorithm which could eliminate the instability found in current methods to solve index 3 DAEs, none of these index reduction methods would be necessary or desirable. Since they require index reduction, the natural index 3 system requires reformulation to a lower index which complicates generalization to existing finite element codes as well as being computationally more expensive.

3.3 Research Goals

In recent years there has been a variety of authors publishing papers on the application well known structural dynamics algorithms to multibody dynamic simulations [39], [3], [40], some of which attempt to leave the DAE in its natural index 3 form as well as include the capability of flexible elements. Such is the goal of the current research: to utilize the GSSSS family of algorithms to search for a method which can find a stable solution to the naturally index 3 DAEs encountered in MBD problems, including any combination of flexible finite elements and rigid multibody elements/joints. Optimally the method would include conservation properties such an energy and momentum, as well as the option to utilize

controllably numerical dissipation if so desired.

3.4 Methodology

In order to determine whether the GSSSS family contains an algorithm capable of overcoming the issues encountered historically by other researchers in the area, the fundamental concepts developed in previous work [28] on structural dynamics applications was leveraged to extend the GSSSS family to cover not only ODEs but DAEs as well. Deep understanding of the subtle issues encountered when extending the linear parent algorithm to nonlinear applications as well as the all-encompassing nature of the GSSSS family of algorithms was expected to provide, at the very least, a fresh perspective on the current state of research in the area.

3.5 Implementation

We have seen in the previous chapter that the method of extending the parent linear time integration algorithm to nonlinear problems can drastically change the performance of the algorithm, as well as provide certain properties such as phase space preservation or energy conservation. To properly assess the extension of the GSSSS family of algorithms to nonlinear differential-algebraic equations it is necessary to test each of the possible implementation options in search of algorithms which may be capable of providing a stable solution to the previously numerically dubious index 3 DAEs. Though something of a brute force search technique, the almost exclusively highly nonlinear multibody systems encountered in real-world applications leave little room for analytic mathematical proofs.

The semi-discrete equations of motion for constrained dynamics may be developed much the say way as in structural dynamics, but this time with the addition of the Lagrange equation to incorporate the constraint equations into the Lagrangian. In structural dynamics we, thanks to Lagrange, define $L = T - V$, where T is the kinetic energy and V is the potential energy of a conservative

system. For constrained dynamics we include the product of the Lagrange multipliers and their corresponding constraint when minimizing the variation of the L. Thus instead of using L [32], alternatively, we employ the descriptive function, $E = T + V + \Phi_i \lambda_i$, the total energy and the corresponding framework. This has improved physical interpretation and is very natural for deriving time stepping schemes with physical insight. Thus the semi-discrete equations of motion defined by the standard:

$$\frac{d}{dt}\left(\frac{\partial E}{\partial \dot{u}_i}\right) + \frac{\partial E}{\partial u_i} = 0 \tag{3.1}$$

now include the product of the vector of Lagrange multipliers and the jacobian of the constrain matrix. Defining $\frac{\partial \Phi}{\partial u} = \mathbf{B}(\mathbf{u})$ the resulting semi-discrete equation of motion can be written as:

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{P}(\mathbf{u}) + \mathbf{B}^T(\mathbf{u})\boldsymbol{\lambda} = \mathbf{F} \tag{3.2}$$

subject to the constraint equations:

$$\boldsymbol{\Phi}(\mathbf{u}, t) = 0 \tag{3.3}$$

Similarly, rewritten as in index 2 DAE we have:

$$\begin{aligned} \mathbf{M}\ddot{\mathbf{u}} + \mathbf{P}(\mathbf{u}) + \mathbf{B}^T(\mathbf{u})\boldsymbol{\lambda} &= \mathbf{F} \\ \boldsymbol{\Phi}_{\mathbf{u}}(\mathbf{u}, t)\dot{\mathbf{u}} + \boldsymbol{\Phi}_t(\mathbf{u}, t) &= 0 \end{aligned} \tag{3.4}$$

And finally in index 1 form:

$$\begin{aligned} \mathbf{M}\ddot{\mathbf{u}} + \mathbf{P}(\mathbf{u}) + \mathbf{B}^T(\mathbf{u})\boldsymbol{\lambda} &= \mathbf{F} \\ \boldsymbol{\Phi}_{\mathbf{u}}(\mathbf{u}, t)\ddot{\mathbf{u}} + (\boldsymbol{\Phi}_{\mathbf{u}}(\mathbf{u}, t)\dot{\mathbf{u}})_q\dot{\mathbf{u}} + 2\boldsymbol{\Phi}_{\mathbf{q}t}(\mathbf{u}, t)\dot{\mathbf{u}} + \boldsymbol{\Phi}_{tt}(\mathbf{u}, t) &= 0 \end{aligned} \tag{3.5}$$

The previous chapter has shown that the time discretization of the $\mathbf{P}(\mathbf{u})$ term, with careful definition of the strain approximation, can result in an energy conserving method. The time discretization of the constraint force, the $\mathbf{B}^T(\mathbf{u})\boldsymbol{\lambda}$ term,

when viewed in the light of the equation of motion time level can be readily seen to take four distinct forms. The equation of motion time level dictates that the each term needs to be evaluated at a consistent time level, namely for the GSSSS family the time t_{n+W_1}. If, like the "classical method" detailed by Masuri [15], the entire term is approximated independently the resulting time discretization follows a "parameters outside" approach. To clarify the difference between a "parameters inside" and a "parameters outside" approach, consider for example the internal force vector $\mathbf{P}(\mathbf{u})$. Knowing that the term needs to be evaluated at the time level t_{n+W_1}, two options are available.

$$\mathbf{P}(\mathbf{u}) = (1 - W_1)\mathbf{P_n}(\mathbf{u_n}) + (W_1)\mathbf{P_{n+1}}(\mathbf{u_{n+1}}) \tag{3.6}$$

$$\mathbf{P}(\mathbf{u}) = \mathbf{P}((1 - W_1)\mathbf{u_n} + (W_1)\mathbf{u_{n+1}}) \tag{3.7}$$

In Eq. 3.6 it can be seen that the time approximation parameter W_1 is outside of the function being evaluated, $\mathbf{P}(\mathbf{u})$, and is therefore termed the parameters outside approach. Conversely, Eq. 3.7 shows the time approximation parameter inside the function, hence the term parameters inside.

An early discussion of the properties of evaluating the time approximation parameters inside the operator or outside of the operator can be found in [41]. More recently, Masuri has shown that the parameters inside approach leads, for most nonlinear structural dynamics applications, to a significantly more numerically robust algorithm. Much in the same way, the goal of this research was to find the algorithms which provide the most robust numerical solutions to the nonlinear DAEs of multibody dynamics. In order to consider all possibilities, the number of ways of approximating the additional $\mathbf{B}^T(\mathbf{u})\boldsymbol{\lambda}$ term is four. For ease of discussion, these four combinations of parameters inside/outside will be referred to as option 1, 2, 3, and 4.

3.5.1 Option 1

Option 1 refers to placing the time approximation parameters outside of both functions $\mathbf{B}^T(\mathbf{u})$ and $\boldsymbol{\lambda}$. This leads to a fully discrete equation of motion containing the constraint force in the form:

$$\widetilde{\mathbf{B}^T\boldsymbol{\lambda}} = (1 - W_1)\mathbf{B}^T(\mathbf{u}_n)\boldsymbol{\lambda}_n + (W_1)\mathbf{B}^T(\mathbf{u}_{n+1})\boldsymbol{\lambda}_{n+1} \tag{3.8}$$

Here we see linear interpolation of the constraint force between time t_n and t_{n+1}.

3.5.2 Option 2

Option 2 is essentially the opposite of option 1: the time approximation parameters are placed inside of both functions. The constraint force now takes the form:

$$\widetilde{\mathbf{B}^T\boldsymbol{\lambda}} = \mathbf{B}^T(\mathbf{u}_{n+W_1})\boldsymbol{\lambda}_{n+W_1} \tag{3.9}$$

In this case the algebraic multiplier λ is no longer linearly approximated between t_n and t_{n+1}, instead the value from the previous time step is unneeded (other than as an initial guess for the nonlinear iteration) and algorithm simply solves directly for the multiplier at time t_{n+W_1}. The $\mathbf{B}$ matrix is evaluated using the same $n + W_1$ time level approximation for $\mathbf{u}$ as all other terms in the fully discrete equation, which of course is dependent on algorithm choice.

3.5.3 Option 3

Option 3 and 4 are now just a mix of the previous two options. Instead of treating the parameters as inside/outside the same for both functions, now one is chosen as parameters inside and the other outside. For option 3 the $\mathbf{B}$ matrix has the parameters outside, whereas the $\boldsymbol{\lambda}$ term is treated in a parameters inside fashion. This mixed form produces a constraint force in the equation of motion as follows:

$$\widetilde{\mathbf{B}^T\boldsymbol{\lambda}} = \left[(1-W_1)\mathbf{B}^T(\mathbf{u}_n) + (W_1)\mathbf{B}^T(\mathbf{u}_{n+1})\right]\boldsymbol{\lambda}_{n+W_1} \tag{3.10}$$

In this particular case the jacobian of the constraints is taken as a linear approximation between time t_n and t_{n+1} whereas, like in option 2, the Lagrange multipliers are solved directly at t_{n+W_1}.

3.5.4 Option 4

The final option, option 4, is simply a reversal of the mixing of option 3. Here the $\mathbf{B}$ matrix has the parameters inside and the $\boldsymbol{\lambda}$ term outside.

$$\widetilde{\mathbf{B}^T\boldsymbol{\lambda}} = \mathbf{B}^T(\mathbf{u}_{n+W_1})\left[(1-W_1)\boldsymbol{\lambda}_n + W_1\boldsymbol{\lambda}_{n+1}\right] \tag{3.11}$$

3.6 GSSSS for Constrained Dynamics in Single Field / Two Field Form

With each of the four options for the additional constraint force term defined, the GSSSS family of algorithms for constrained dynamics is as follows.

Nonlinear Scheme in Single Field Form Unified Predictor Multi-Corrector Representation

The effective DAE employing the normalized time weighted residual approach is given by

$$\mathbf{M}\widetilde{\ddot{\mathbf{u}}} + \mathbf{P}(\widetilde{\mathbf{u}}) + \widetilde{\mathbf{B}^T\boldsymbol{\lambda}} = \widetilde{\mathbf{F}} \tag{3.12}$$

$$\chi_{Cu}\boldsymbol{\Phi}(\mathbf{u}_{n+1}) = 0 \tag{3.13}$$

Notice that the constraint equation has been scaled by χ_{Cu} in attempt to avoid ill-conditioning of the jacobian matrix, similar to that described in [40].

The generic $\widetilde{\mathbf{B}^T\boldsymbol{\lambda}}$ term takes the specific form corresponding to the option of choice, defined by Eqs. 3.8-3.11.

Employ the Newton-Raphson method to iteratively solve for the nonlinear effective DAE above:

At the beginning of time step, predict the state vectors

$$\mathbf{u}_{n+1}^{k} = \chi_{Pu}^{(1)}\ \mathbf{u}_n + \chi_{Pu}^{(2)}\ \dot{\mathbf{u}}_n + \chi_{Pu}^{(3)}\ \ddot{\mathbf{u}}_n \tag{3.14}$$

$$\widetilde{\mathbf{u}}_{n+1}^{k} = \chi_{P\tilde{u}}^{(1)}\ \mathbf{u}_n + \chi_{P\tilde{u}}^{(2)}\ \dot{\mathbf{u}}_n + \chi_{P\tilde{u}}^{(3)}\ \ddot{\mathbf{u}}_n \tag{3.15}$$

$$\widetilde{\dot{\mathbf{u}}}_{n+1}^{k} = \chi_{P\tilde{v}}^{(1)}\ \mathbf{u}_n + \chi_{P\tilde{v}}^{(2)}\ \dot{\mathbf{u}}_n + \chi_{P\tilde{v}}^{(3)}\ \ddot{\mathbf{u}}_n \tag{3.16}$$

$$\widetilde{\ddot{\mathbf{u}}}_{n+1}^{k} = \chi_{P\tilde{a}}^{(1)}\ \mathbf{u}_n + \chi_{P\tilde{a}}^{(2)}\ \dot{\mathbf{u}}_n + \chi_{P\tilde{a}}^{(3)}\ \ddot{\mathbf{u}}_n \tag{3.17}$$

$$\widetilde{\boldsymbol{\lambda}}_{n+1}^{k} = \boldsymbol{\lambda}_n \tag{3.18}$$

Start nonlinear iteration. Solve for $\Delta\boldsymbol{\delta}_{n+1}^{k+1}$ and $\Delta\boldsymbol{\lambda}_{n+1}^{k+1}$ from

$$\mathbf{J}\begin{bmatrix}\Delta\boldsymbol{\delta}_{n+1}^{k+1}\\ \Delta\boldsymbol{\lambda}_{n+1}^{k+1}\end{bmatrix} = \begin{bmatrix}-\mathbf{e_1}\\ -\mathbf{e_2}\end{bmatrix} \tag{3.19}$$

where

$$\mathbf{e_1} = \mathbf{M}\widetilde{\ddot{\mathbf{u}}} + \mathbf{P}(\widetilde{\mathbf{u}}) + \widetilde{\mathbf{B}^T\boldsymbol{\lambda}} - \widetilde{\mathbf{F}} \tag{3.20}$$

$$\mathbf{e_2} = \chi_{Cu}\boldsymbol{\Phi}(\mathbf{u}_{n+1}) \tag{3.21}$$

and

$$\mathbf{J} = \begin{bmatrix}\frac{\partial \mathbf{e_1}}{\partial \mathbf{u^*}_{n+1}} & \frac{\partial \mathbf{e_1}}{\partial \boldsymbol{\lambda}_{n+1}}\\ \mathbf{B}(\mathbf{u}_{n+1}) & \mathbf{0}\end{bmatrix} \tag{3.22}$$

with $\mathbf{u^*}_{n+1} = \mathbf{u}_{n+1}$, $\dot{\mathbf{u}}_{n+1}$, or $\ddot{\mathbf{u}}_{n+1}$ for d-, v-, or a-form, respectively.

Then correct the primary variables as follows:

$$\mathbf{u}_{n+1}^{k+1} = \mathbf{u}_{n+1}^{k} + \chi_{Cu}\Delta\boldsymbol{\delta}_{n+1}^{k+1} \tag{3.23}$$

$$\widetilde{\mathbf{u}}_{n+1}^{k+1} = \widetilde{\mathbf{u}}_{n+1}^{k} + \chi_{C\tilde{u}}\Delta\boldsymbol{\delta}_{n+1}^{k+1} \tag{3.24}$$

$$\widetilde{\dot{\mathbf{u}}}_{n+1}^{k+1} = \widetilde{\dot{\mathbf{u}}}_{n+1}^{k} + \chi_{C\tilde{v}}\Delta\boldsymbol{\delta}_{n+1}^{k+1} \tag{3.25}$$

$$\widetilde{\ddot{\mathbf{u}}}_{n+1}^{k+1} = \widetilde{\ddot{\mathbf{u}}}_{n+1}^{k} + \chi_{C\tilde{a}}\Delta\boldsymbol{\delta}_{n+1}^{k+1} \tag{3.26}$$

$$\boldsymbol{\lambda}_{n+1}^{k+1} = \boldsymbol{\lambda}_{n+1}^{k} + \Delta\boldsymbol{\lambda}_{n+1}^{k+1} \tag{3.27}$$

Repeat nonlinear iterations until the solution converges. Once a converged solution is found, update remaining variables as:

$$\ddot{\mathbf{u}}_{n+1} = \ddot{\mathbf{u}}_{n} + (\tilde{\ddot{\mathbf{u}}}_{n+1}^{k+1} - \ddot{\mathbf{u}}_{n})/(\Lambda_6 W_1) \tag{3.28}$$

$$\dot{\mathbf{u}}_{n+1} = \dot{\mathbf{u}}_{n} + \lambda_4\ddot{\mathbf{u}}_{n}\Delta t + \lambda_5(\ddot{\mathbf{u}}_{n+1} - \ddot{\mathbf{u}}_{n})\Delta t \tag{3.29}$$

$$\mathbf{u}_{n+1} = \mathbf{u}_{n} + \lambda_1\dot{\mathbf{u}}_{n}\Delta t + \lambda_2\ddot{\mathbf{u}}_{n}\Delta t^2 + \lambda_3(\ddot{\mathbf{u}}_{n+1} - \ddot{\mathbf{u}}_{n})\Delta t^2 \tag{3.30}$$

Notice that the scalar λ_i values in the updates are the algorithm specific parameters resulting from the GSSSS framework and are in no way associated with the bold vector of Lagrange multipliers, $\widetilde{\boldsymbol{\lambda}}_{n+1}^{k+1}$.

As a result of previous work [27], it is known that the value of acceleration that results from the algorithm is not always at time t_{n+1}. To obtain second order accurate accelerations the time level of the acceleration must be accounted for. Much in the same way, the time level of the computed Lagrange multipliers in the algorithm is t_{n+W_1}. In order to obtain a consistent convergence plot, the Lagrange multipliers at the final time step may be interpolated forward in time to obtain values at t_{n+1} as follows:

$$\boldsymbol{\lambda}_{n+1} = (1 - (2 - W_1))\boldsymbol{\lambda}_{n} + (2 - W_1)\widetilde{\boldsymbol{\lambda}}_{n+1}^{k+1} \tag{3.31}$$

3.7 Energy Conservation

As mentioned in the introductory chapter it has been recently shown in a series of papers by Betsch et al., starting specifically in [17], that an algorithm which

conserves energy for structural dynamic applications maintains that energy conserving property so long as the jacobian of the constraints $\mathbf{B}$ is evaluated at the midpoint configuration. Specifically, to maintain the energy conservation property of an algorithm upon its extension from ODEs to DAEs we must evaluate $\mathbf{B}(u_{n+\frac{1}{2}})$. This is not only consistent with the requirements understood from energy conservative algorithms for structural dynamics, but also satisfies the requirement that the equation of motion be satisfied at a *consistent* time level. As a result, the energy conserving algorithm described in the previous chapter for geometrically nonlinear truss elements will maintain the conservation property so long as the additional $\widetilde{\mathbf{B}^T\boldsymbol{\lambda}}$ term is equivalent to $\mathbf{B}(u_{n+\frac{1}{2}})$. It is readily seen that only option 2 and option 4 guarantee such equivalence in the case of nonlinear equations when using the U0(1,1,1) or V0(1,1,$\rho_{3\infty}$) algorithms.

In the following chapter a variety of rigid and flexible problems are described in the search for a non-dissipative algorithm which is capable of obtaining a stable solution for index 3 multibody dynamics. The search space includes the three non-dissipative algorithms contained within the GSSSS framework: the Newmark method (U0(1,1,0)), the well known Midpoint rule with endpoint acceleration (U0/V0(1,1,1)), and the lesser known Midpoint rule with midpoint acceleration (V0(1,1,0)) in conjunction with any of the options 1 through 4 described above. These three algorithms and four options result in 12 distinct algorithmic implementations. Of these, it is the understanding of the author that only two of these methods have been investigated previously in the literature: the Newmark method using option 1, and the Midpoint rule with endpoint acceleration using option 1. The V0 family and the normalized weighted residual procedure are not well know to the community, making it highly unlikely the remaining 10 algorithms have been the focus of any prior research. It was this vast portion of the solution space which had not previously been investigated which fostered optimism that somewhere within the GSSSS family of algorithms a truly optimal algorithm for directly integrating index 3 differential-algebraic equations would be found.

3.8 Two Field Form Equivalence

Though the main focus of this research is upon the natural single field form of the GSSSS family of algorithms, it is noteworthy to point out the equivalence of the V0(1,1,0) algorithm to existing two field form energy conserving algorithms. Gonzalez [42] introduced an energy conserving technique for differential-algebraic equations based on a two field approach. In this method a discrete derivative operator is applied to guarantee energy conservation of the internal force and constraint force terms. Betsch [17] notes that in the method of Gonzalez, for systems in which the potential energy, kinetic energy, and constraints are at most quadratic, the discrete derivate can be replaced by midpoint evaluation of the quantities within the equation of motion. The resulting algorithms appears below.

$$\mathbf{u}_{n+1} - \mathbf{u}_n = \frac{\Delta t}{2}(\mathbf{v}_{n+1} + \mathbf{v}_n) \tag{3.32}$$

$$\mathbf{M}\frac{\mathbf{v}_{n+1} - \mathbf{v}_n}{\Delta t} + \nabla V(\mathbf{u}_{n+\frac{1}{2}}) + \mathbf{B}^T(\mathbf{u}_{n+\frac{1}{2}})\boldsymbol{\lambda}_{n+1} = 0 \tag{3.33}$$

Here V is the strain energy of the material model and $\mathbf{u}_{n+\frac{1}{2}}$ is defined simply by $\frac{\mathbf{u}_{n+1}+\mathbf{u}_n}{2}$.

To see that this method is mathematically equivalent to the single field form GSSSS algorithm V0(1,1,0) we rewrite the V0 family of algorithms (Algorithm 2 from the previous chapter) excluding the external force and damping terms. The parameters which define the algorithm are:

$$W_1 = W_2 = W_3 = \frac{1}{2} \tag{3.34}$$
$$\Lambda_1 = \Lambda_4 = \Lambda_5 = 1 \tag{3.35}$$
$$\Lambda_2 = \Lambda_3 = \frac{1}{2} \tag{3.36}$$
$$\Lambda_6 = 2 \tag{3.37}$$
$$\lambda_1 = \lambda_4 = \lambda_5 = 1 \tag{3.38}$$
$$\lambda_2 = \lambda_3 = \frac{1}{2} \tag{3.39}$$
$$\tag{3.40}$$

The V0(1,1,0) algorithm is then:

$$\left(\mathbf{M} + \frac{\Delta t^2}{4}\mathbf{K}\right)\Delta\mathbf{a} = -\mathbf{M}\ddot{\mathbf{u}}_n - \mathbf{K}\left(\mathbf{u}_n + \frac{1}{2}\dot{\mathbf{u}}_n\Delta t + \frac{1}{4}\ddot{\mathbf{u}}_n\Delta t^2\right) \tag{3.41}$$
$$\mathbf{u}_{n+1} = \mathbf{u}_n + \dot{\mathbf{u}}_n\Delta t + \frac{1}{2}\ddot{\mathbf{u}}_n\Delta t^2 + \frac{1}{2}\Delta\mathbf{a}\Delta t^2 \tag{3.42}$$
$$\dot{\mathbf{u}}_{n+1} = \dot{\mathbf{u}}_n + \ddot{\mathbf{u}}_n\Delta t + \Delta\mathbf{a}\Delta t \tag{3.43}$$
$$\ddot{\mathbf{u}}_{n+1} = \ddot{\mathbf{u}}_n + \Delta\mathbf{a} \tag{3.44}$$

Rearranging the updates for the velocity (Eq. 3.43) and displacement (Eq. 3.42) by substituting Eq. 3.44 results in the relations:

$$\mathbf{a}_{n+1} = \frac{\mathbf{v}_{n+1} - \mathbf{v}_n}{\Delta t} \tag{3.45}$$
$$\mathbf{a}_{n+1} = \frac{2}{\Delta t}(\mathbf{u}_{n+1} - \mathbf{u}_n - \mathbf{v}_n\Delta t) \tag{3.46}$$

Substituting these relations for $\mathbf{a}_{n+1}$ back into 3.41 yields:

$$\mathbf{M}\frac{\mathbf{v}_{n+1} - \mathbf{v}_n}{\Delta t} + K\mathbf{u}_{n+\frac{1}{2}} + \mathbf{B}^T(\mathbf{u}_{n+\frac{1}{2}})\boldsymbol{\lambda}_{n+1} = 0 \tag{3.47}$$

or, in the case of nonlinear internal force:

$$\mathbf{M}\frac{\mathbf{v}_{n+1}-\mathbf{v}_n}{\Delta t}+\nabla V(\mathbf{u}_{n+\frac{1}{2}})+\mathbf{B}^T(\mathbf{u}_{n+\frac{1}{2}})\boldsymbol{\lambda}_{n+1}=0 \tag{3.48}$$

It is clear after this algebraic manipulation that the V0(1,1,0) algorithm is precisely equivalent to the two field form energy conserving algorithm of Gonzalez (Eq. 3.33) with the simple difference that the GSSSS (single field form) version replaces $\frac{\mathbf{v}_{n+1}-\mathbf{v}_n}{\Delta t}$ with the term $\mathbf{a}_{n+1}$. Notable is that, as shown in previous work, the equation of motion time level theory proves that the term $\mathbf{a}_{n+1}$ is the value of acceleration at the time point t_{n+W_1}, or in this case, $t_{n+\frac{1}{2}}$. As such, the V0(1,1,0) algorithm is termed the midpoint rule (equation of motion satisfied at exactly $t_{n+\frac{1}{2}}$) with midpoint acceleration (accelerations not updated to t_{n+1}).

The equivalence of the single and two field form is independent of the method of extension to nonlinear constrained equations. The treatment of the addition term $\mathbf{B}^T\boldsymbol{\lambda}$ may follow any option 1, 2, 3, or 4 in either case. The results in the following chapter focus on the single field form, but results of the V0(1,1,0) algorithm are identical in all cases to the two field form methodology described above.

Nonlinear Scheme in Two Field v-Form Representation

The fully discrete DAE is given by

$$\mathbf{M}\frac{\mathbf{v}_{n+1}-\mathbf{v}_n}{\Delta t}+\nabla V(\mathbf{u}_{n+\frac{1}{2}})+\mathbf{B}^T(\mathbf{u}_{n+\frac{1}{2}})\boldsymbol{\lambda}_{n+1}=0 \tag{3.49}$$

$$\frac{2}{\Delta t}\boldsymbol{\Phi}(\mathbf{u}_{n+1})=0 \tag{3.50}$$

Notice that the constraint equation has been scaled by $\frac{2}{\Delta t}$ in attempt to avoid ill-conditioning of the jacobian matrix, similar to that described in [40]. The generic $\widetilde{\mathbf{B}^T\boldsymbol{\lambda}}$ term takes the specific form corresponding to the option of choice, defined by Eqs. 3.8-3.11.

Employ the Newton-Raphson method to iteratively solve for the nonlinear effective DAE above:

At the beginning of time step, predict the state vectors

$$\dot{\mathbf{u}}_{n+1}^{k} = \dot{\mathbf{u}}_n \tag{3.51}$$

$$\widetilde{\boldsymbol{\lambda}}_{n+1}^{k} = \boldsymbol{\lambda}_n \tag{3.52}$$

$$\mathbf{u}_{n+\frac{1}{2}} = \mathbf{u}_n + \frac{\Delta t}{4}(\dot{\mathbf{u}}_{n+1} + \dot{\mathbf{u}}_n) \tag{3.53}$$

Start nonlinear iteration. Solve for $\Delta\dot{\mathbf{u}}_{n+1}^{k+1}$ and $\Delta\boldsymbol{\lambda}_{n+1}^{k+1}$ from

$$\mathbf{J}\begin{bmatrix}\Delta\dot{\mathbf{u}}_{n+1}^{k+1} \\ \Delta\boldsymbol{\lambda}_{n+1}^{k+1}\end{bmatrix} = \begin{bmatrix}-\mathbf{e_1} \\ -\mathbf{e_2}\end{bmatrix} \tag{3.54}$$

where

$$\mathbf{e_1} = \mathbf{M}\frac{\mathbf{v}_{n+1} - \mathbf{v}_n}{\Delta t} + \nabla V(\mathbf{u}_{n+\frac{1}{2}}) + \mathbf{B}^T(\mathbf{u}_{n+\frac{1}{2}})\boldsymbol{\lambda}_{n+1} - \widetilde{\mathbf{F}} \tag{3.55}$$

$$\mathbf{e_2} = \frac{2}{\Delta t}\boldsymbol{\Phi}(\mathbf{u}_{n+1}) \tag{3.56}$$

and

$$\mathbf{J} = \begin{bmatrix}\frac{\partial \mathbf{e_1}}{\partial \dot{\mathbf{u}}_{n+1}} & \frac{\partial \mathbf{e_1}}{\partial \boldsymbol{\lambda}_{n+1}} \\ \mathbf{B}(\mathbf{u}_{n+1}) & \mathbf{0}\end{bmatrix} \tag{3.57}$$

Then correct the primary variables as follows:

$$\dot{\mathbf{u}}_{n+1}^{k+1} = \dot{\mathbf{u}}_{n+1}^{k} + \Delta\dot{\mathbf{u}}_{n+1}^{k+1} \tag{3.58}$$

$$\boldsymbol{\lambda}_{n+1}^{k+1} = \boldsymbol{\lambda}_{n+1}^{k} + \Delta\boldsymbol{\lambda}_{n+1}^{k+1} \qquad \mathbf{u}_{n+\frac{1}{2}} = \mathbf{u}_n + \frac{\Delta t}{4}(\dot{\mathbf{u}}_{n+1} + \dot{\mathbf{u}}_n) \tag{3.59}$$

Repeat nonlinear iterations until the solution converges. Once a converged solution is found, update displacements as:

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \frac{\Delta t}{2}(\dot{\mathbf{u}}_{n+1} + \dot{\mathbf{u}}_n) \tag{3.60}$$

Chapter 4

Numerical Examples

4.1 Rigid Multibody Dynamics

4.1.1 Double Pendulum

Demonstration of the four combinations of parameters inside/outside on the $\mathbf{B}^T\lambda$ term is first shown for the rigid double pendulum shown in Fig 4.1. The model consists of two planar rigid bars, each with a point mass at its end, pinned to the ground at the origin. Gravity is the only external force acting on the system. Redundant coordinates (x, y, θ) are used for simplicity to track the location of each point mass during the simulation. This problem is discussed in great detail in [3], with the pertinent information repeated below.

In general the problem is of the form:

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{B}^T(\mathbf{u})\boldsymbol{\lambda} = \mathbf{F} \tag{4.1}$$

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

The problem consists of six degrees of freedom and four Lagrange multipliers. That is, the unknowns are:

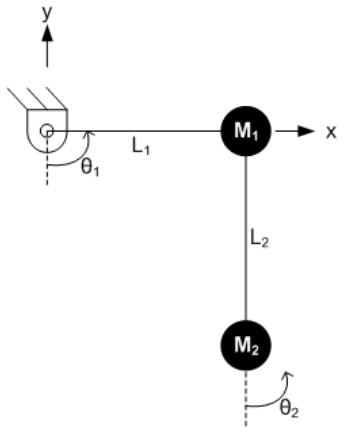


Figure 4.1: Rigid Double Pendulum

$$\mathbf{u} = \begin{bmatrix} x_1 & y_1 & \theta_1 & x_2 & y_2 & \theta_2 \end{bmatrix}^T \tag{4.3}$$

$$\dot{\mathbf{u}} = \begin{bmatrix} \dot{x}_1 & \dot{y}_1 & \dot{\theta}_1 & \dot{x}_2 & \dot{y}_2 & \dot{\theta}_2 \end{bmatrix}^T \tag{4.4}$$

$$\ddot{\mathbf{u}} = \begin{bmatrix} \ddot{x}_1 & \ddot{y}_1 & \ddot{\theta}_1 & \ddot{x}_2 & \ddot{y}_2 & \ddot{\theta}_2 \end{bmatrix}^T \tag{4.5}$$

$$\boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 & \lambda_2 & \lambda_3 & \lambda_4 \end{bmatrix}^T \tag{4.6}$$

Four constraint equations serve to keep the length of both bars constant:

$$\boldsymbol{\Phi} = \begin{bmatrix} x_1 - L_1 sin(\theta_1) \\ y_1 + L_1 cos(\theta_1) \\ x_2 - x_1 - L_2 sin(\theta_2) \\ y_2 - y_1 + L_2 cos(\theta_2) \end{bmatrix} \tag{4.7}$$

The $\mathbf{B}$ matrix follows as:

$$\mathbf{B}(\mathbf{u}) = \frac{\partial \mathbf{\Phi}(\mathbf{u})}{\partial \mathbf{u}} = \begin{bmatrix} 1 & 0 & -L_1 cos(\theta_1) & 0 & 0 & 0 \\ 0 & 1 & -L_1 sin(\theta_1) & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & -L_2 cos(\theta_2) \\ 0 & -1 & 0 & 0 & 1 & -L_2 sin(\theta_2) \end{bmatrix} \tag{4.8}$$

As the θ_i degree of freedom is redundant and used simply to avoid repetitive calls to tan^{-1}, the potential energy (PE), kinetic energy (KE), the mass matrix, and external forces use only the Cartesian coordinates x and y as shown below.

$$KE = \frac{1}{2} M_1 (\dot{x}_1^2 + \dot{y}_1^2) + \frac{1}{2} M_2 (\dot{x}_2^2 + \dot{y}_2^2) \tag{4.9}$$

$$PE = M_1 g y_1 + M_2 g y_2 \tag{4.10}$$

$$\mathbf{M} = diag(\begin{bmatrix} M_1 & M_1 & 0 & M_2 & M_2 & 0 \end{bmatrix}) \tag{4.11}$$

$$\mathbf{F} = \begin{bmatrix} 0 & -M_1 g & 0 & 0 & -M_2 g & 0 \end{bmatrix}^T \tag{4.12}$$

In all simulations below the problem parameters are taken as $M_1 = M_2 = 1$, $L_1 = L_2 = 1$, and the gravitational constant is rounded to $g = 10$. The initial conditions are taken as follows:

$$\mathbf{u_0} = \begin{bmatrix} 1 & 0 & \frac{\pi}{2} & 1 & -1 & 0 \end{bmatrix}^T \tag{4.13}$$

$$\dot{\mathbf{u}}_\mathbf{0} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{4.14}$$

The jacobian matrix under options 1, 2, 3, and 4 for the single field form follow as:

Option 1:

$$\mathbf{J} = \begin{bmatrix} W_1 \Lambda_6 \mathbf{M} + W_1 \lambda_3 \Delta t^2 \mathbf{J}_{B\lambda}(\mathbf{u}_{n+1}, \boldsymbol{\lambda}_{n+1}) & W_1 \mathbf{B}(\mathbf{u}_{n+1})^T \\ \mathbf{B}(\mathbf{u}_{n+1}) & \mathbf{0} \end{bmatrix} \tag{4.15}$$

Option 2:

$$\mathbf{J} = \begin{bmatrix} W_1\Lambda_6\mathbf{M} + \Lambda_3 W_3 \Delta t^2 \mathbf{J}_{B\lambda}(\widetilde{\mathbf{u}}, \boldsymbol{\lambda}_{n+1}) & \mathbf{B}(\widetilde{\mathbf{u}})^T \\ \mathbf{B}(\mathbf{u}_{n+1}) & \mathbf{0} \end{bmatrix} \tag{4.16}$$

Option 3:

$$\mathbf{J} = \begin{bmatrix} W_1\Lambda_6\mathbf{M} + W_1\lambda_3 \Delta t^2 \mathbf{J}_{B\lambda}(\mathbf{u}_{n+1}, \boldsymbol{\lambda}_{n+1}) & (1-W_1)\mathbf{B}(\mathbf{u}_n) + W_1\mathbf{B}(\mathbf{u}_{n+1})^T \\ \mathbf{B}(\mathbf{u}_{n+1}) & \mathbf{0} \end{bmatrix} \tag{4.17}$$

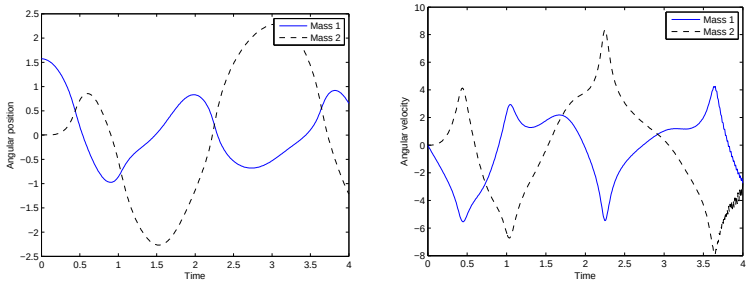
Option 4:

$$\mathbf{J} = \begin{bmatrix} W_1\Lambda_6\mathbf{M} + \Lambda_3 W_3 W_1 \Delta t^2 \mathbf{J}_{B\lambda}(\widetilde{\mathbf{u}}, \boldsymbol{\lambda}_{n+1}) & W_1\mathbf{B}(\widetilde{\mathbf{u}})^T \\ \mathbf{B}(\mathbf{u}_{n+1}) & \mathbf{0} \end{bmatrix} \tag{4.18}$$

where $\mathbf{J}_{B\lambda}(\mathbf{u}, \boldsymbol{\lambda})$ is a 6x6 matrix of zeros with entries:

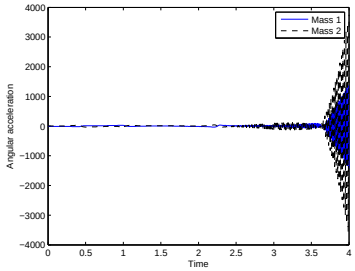
$$\begin{aligned} \mathbf{J}_{B\lambda}(3,3) &= L_1 \sin\theta_1 \boldsymbol{\lambda}_1 - L_1 \cos\theta_1 \boldsymbol{\lambda}_2 \\ \mathbf{J}_{B\lambda}(6,6) &= L_2 \sin\theta_2 \boldsymbol{\lambda}_3 - L_2 \cos\theta_2 \boldsymbol{\lambda}_4 \end{aligned} \tag{4.19}$$

The problem was run using options 1, 2, 3, and 4 for the three non-dissipative algorithms U0(1,1,0), U0/V0(1,1,1), and V0(1,1,0) using the a-form representation of the single field form algorithm. The time step used in all simulations was $\Delta t = 0.01$, the nonlinear iteration tolerance 10^{-8}, and the number of nonlinear iterations limited to 100 per time step. The simulation duration was first set to 10 seconds, though not all algorithms were able to provide a solution. The results of which are shown below in Figs. 4.2-4.13. Those which did yield a stable solution were then tested on runs to 100 seconds to determine long duration stability. A discussion of the results follows the figures below.

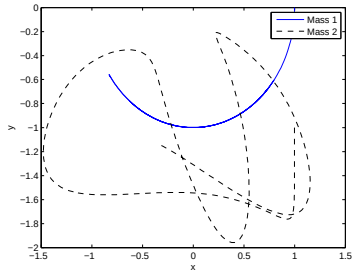


(a) Angular Positions

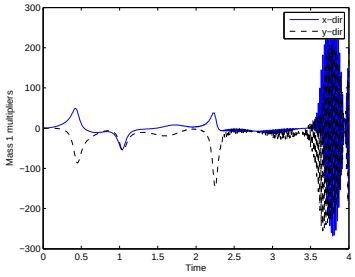
(b) Angular Velocities



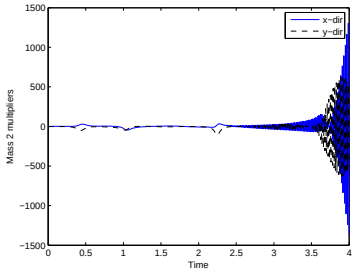
(c) Angular Accelerations



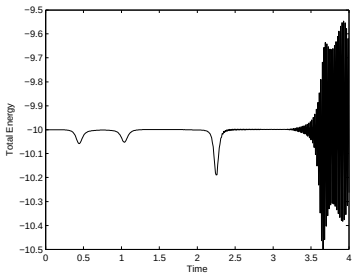
(d) Trajectories



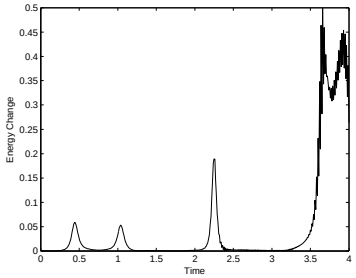
(e) Mass 1 Multipliers



(f) Mass 2 Multipliers

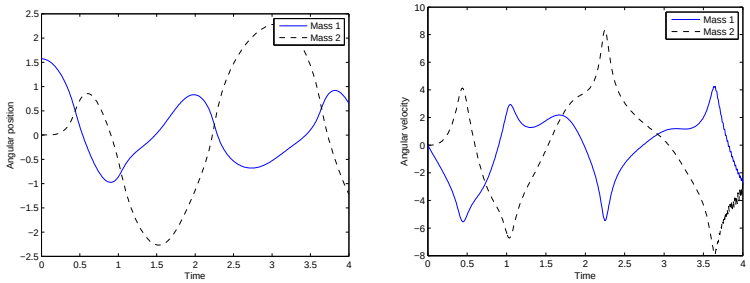


(g) Total Energy



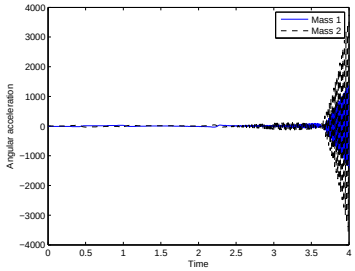
(h) Energy Change

Figure 4.2: Double Pendulum - Option 1 - U0(1,1,0)

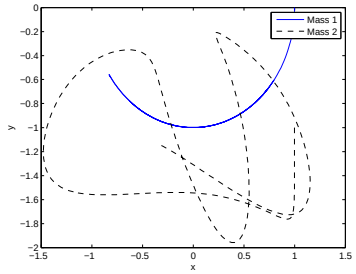


(a) Angular Positions

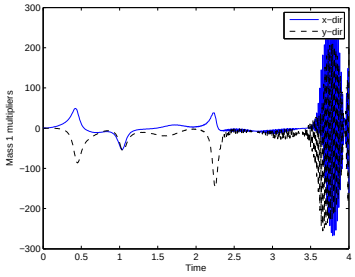
(b) Angular Velocities



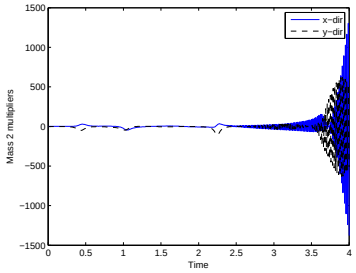
(c) Angular Accelerations



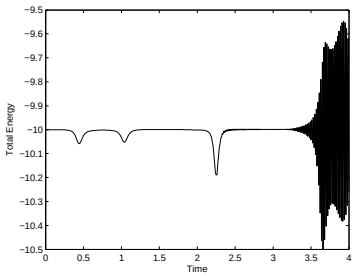
(d) Trajectories



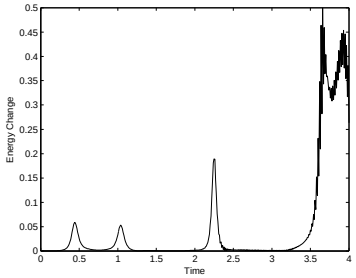
(e) Mass 1 Multipliers



(f) Mass 2 Multipliers

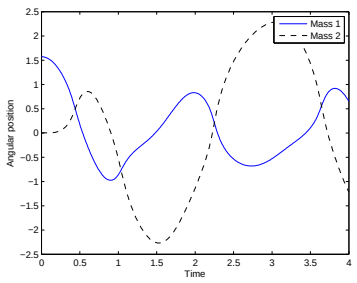


(g) Total Energy

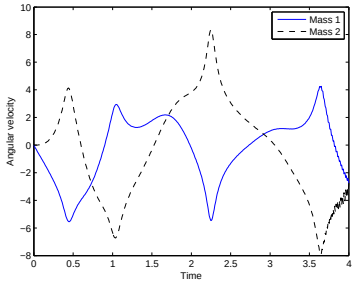


(h) Energy Change

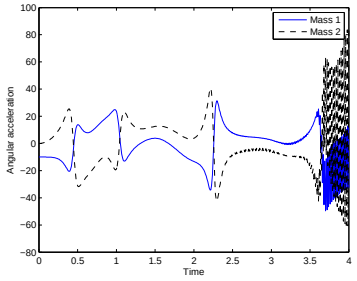
Figure 4.3: Double Pendulum - Option 1 - V0(1,1,1)



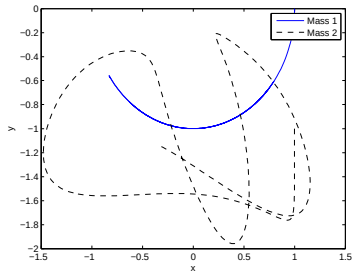
(a) Angular Positions



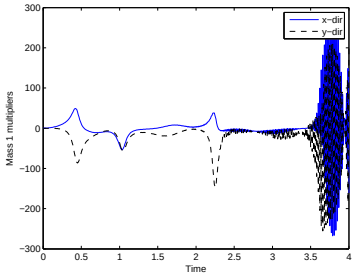
(b) Angular Velocities



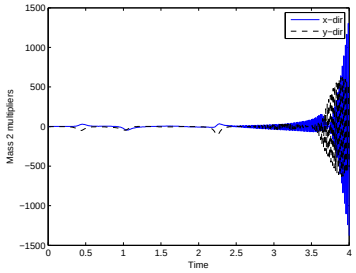
(c) Angular Accelerations



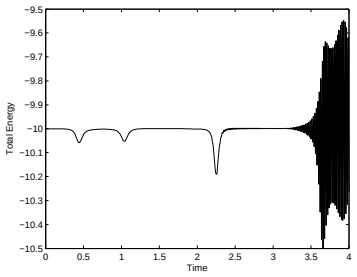
(d) Trajectories



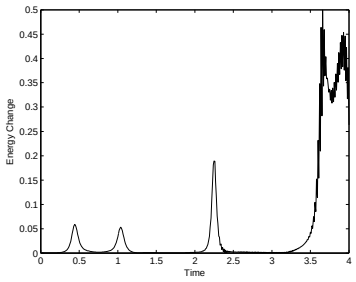
(e) Mass 1 Multipliers



(f) Mass 2 Multipliers

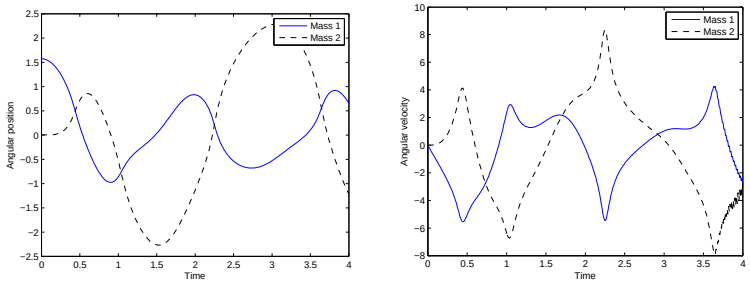


(g) Total Energy



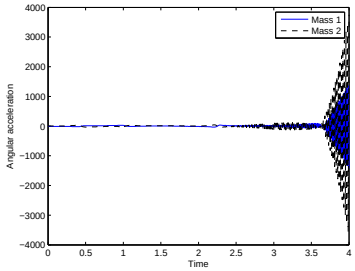
(h) Energy Change

Figure 4.4: Double Pendulum - Option 1 - V0(1,1,0)

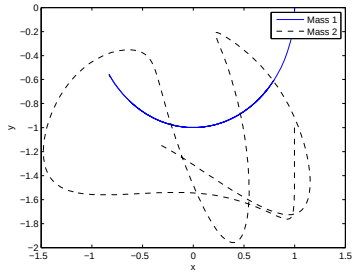


(a) Angular Positions

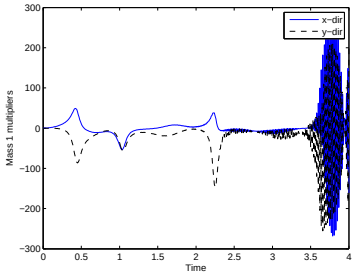
(b) Angular Velocities



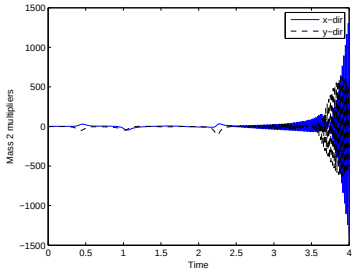
(c) Angular Accelerations



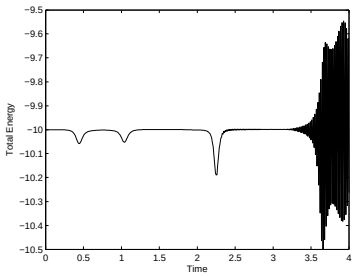
(d) Trajectories



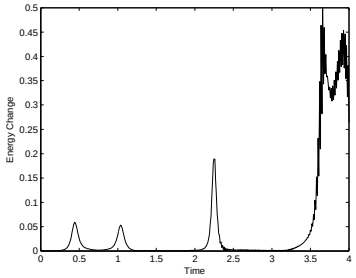
(e) Mass 1 Multipliers



(f) Mass 2 Multipliers

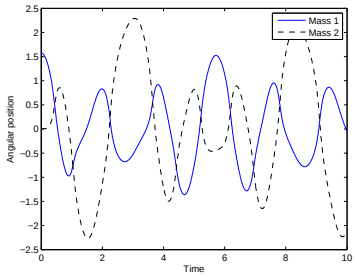


(g) Total Energy

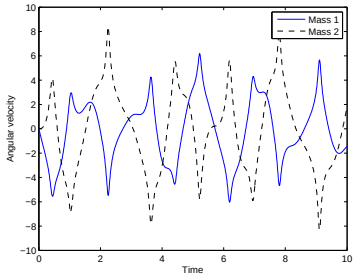


(h) Energy Change

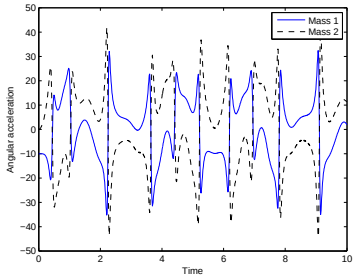
Figure 4.5: Double Pendulum - Option 2 - U0(1,1,0)



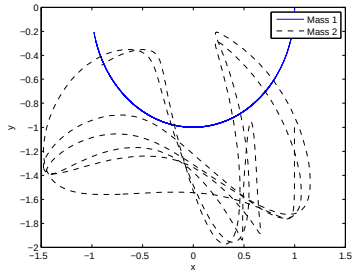
(a) Angular Positions



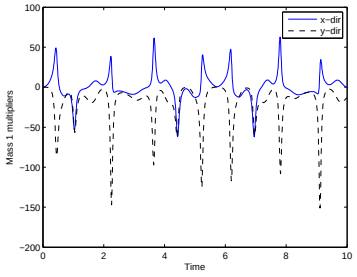
(b) Angular Velocities



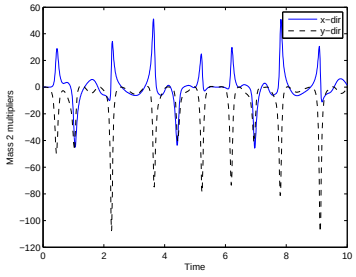
(c) Angular Accelerations



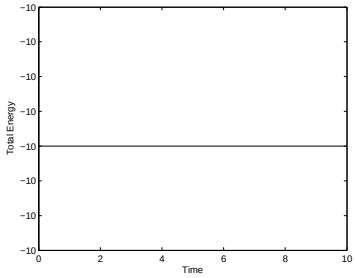
(d) Trajectories



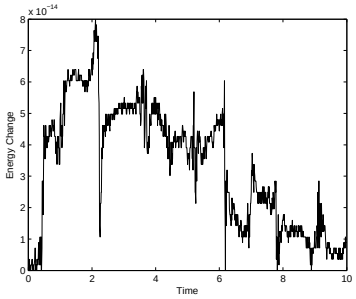
(e) Mass 1 Multipliers



(f) Mass 2 Multipliers

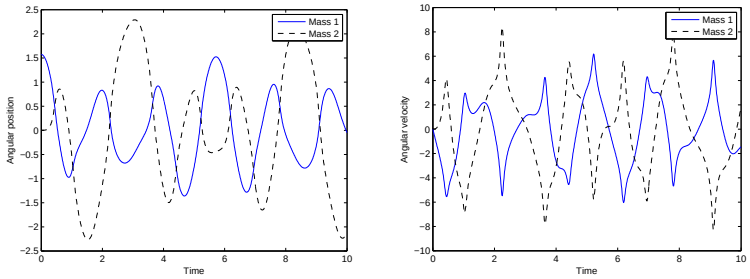


(g) Total Energy



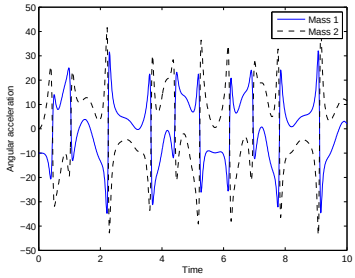
(h) Energy Change

Figure 4.6: Double Pendulum - Option 2 - V0(1,1,1)

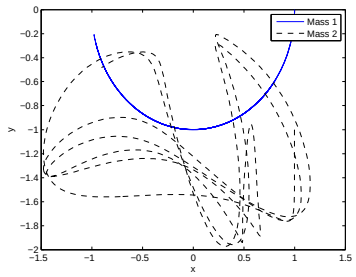


(a) Angular Positions

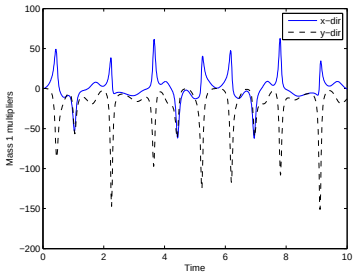
(b) Angular Velocities



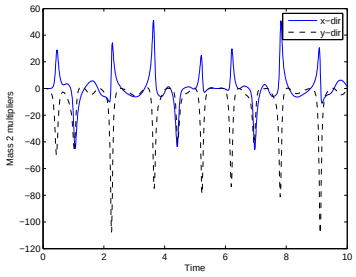
(c) Angular Accelerations



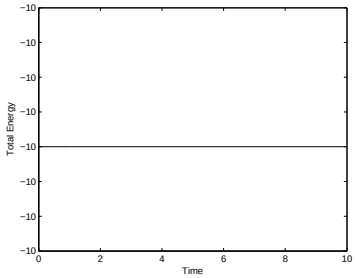
(d) Trajectories



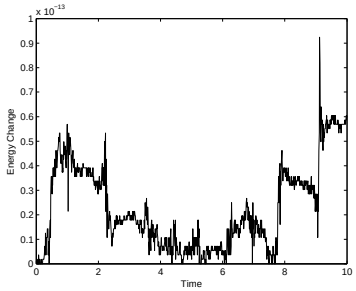
(e) Mass 1 Multipliers



(f) Mass 2 Multipliers

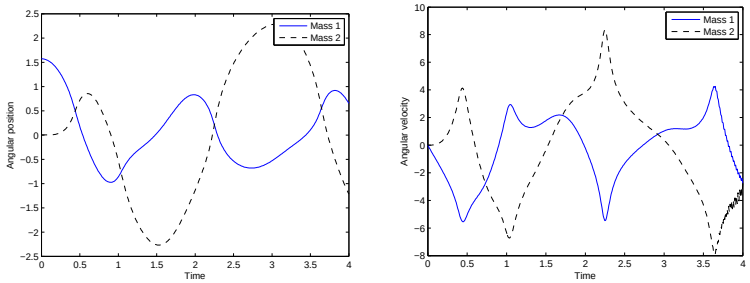


(g) Total Energy



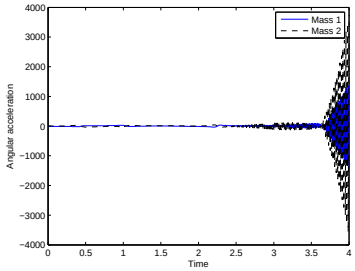
(h) Energy Change

Figure 4.7: Double Pendulum - Option 2 - V0(1,1,0)

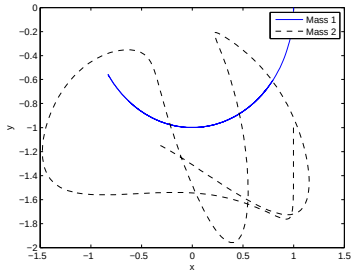


(a) Angular Positions

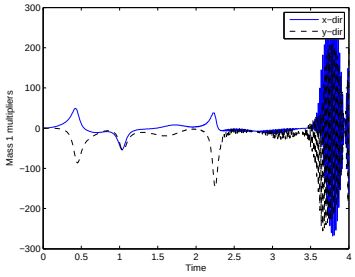
(b) Angular Velocities



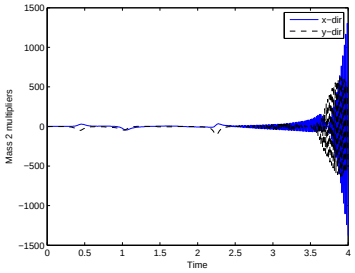
(c) Angular Accelerations



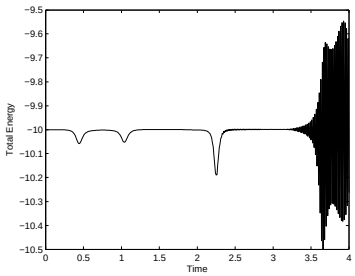
(d) Trajectories



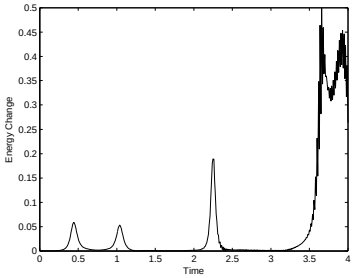
(e) Mass 1 Multipliers



(f) Mass 2 Multipliers

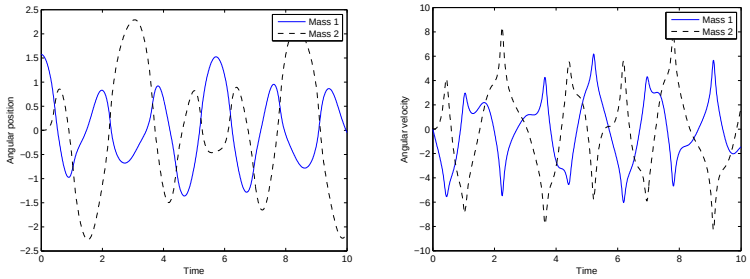


(g) Total Energy



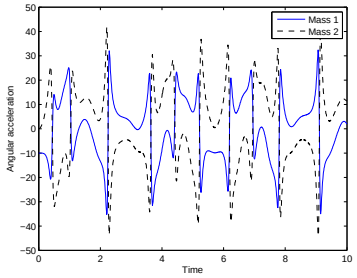
(h) Energy Change

Figure 4.8: Double Pendulum - Option 3 - U0(1,1,0)

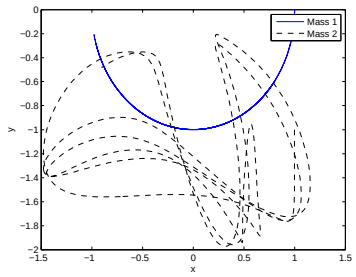


(a) Angular Positions

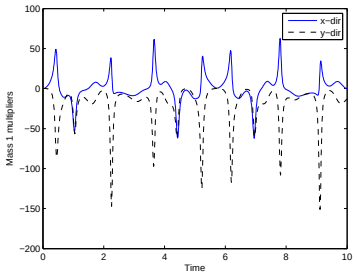
(b) Angular Velocities



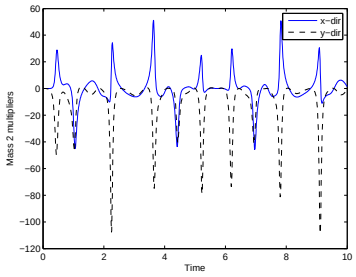
(c) Angular Accelerations



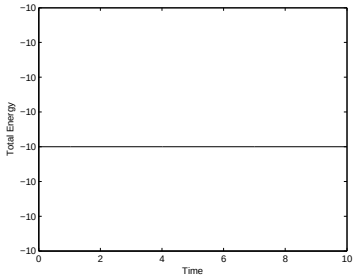
(d) Trajectories



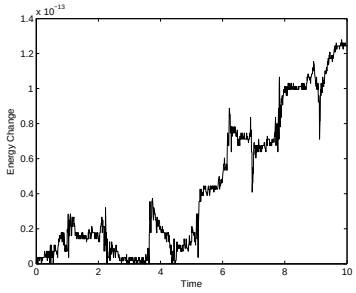
(e) Mass 1 Multipliers



(f) Mass 2 Multipliers

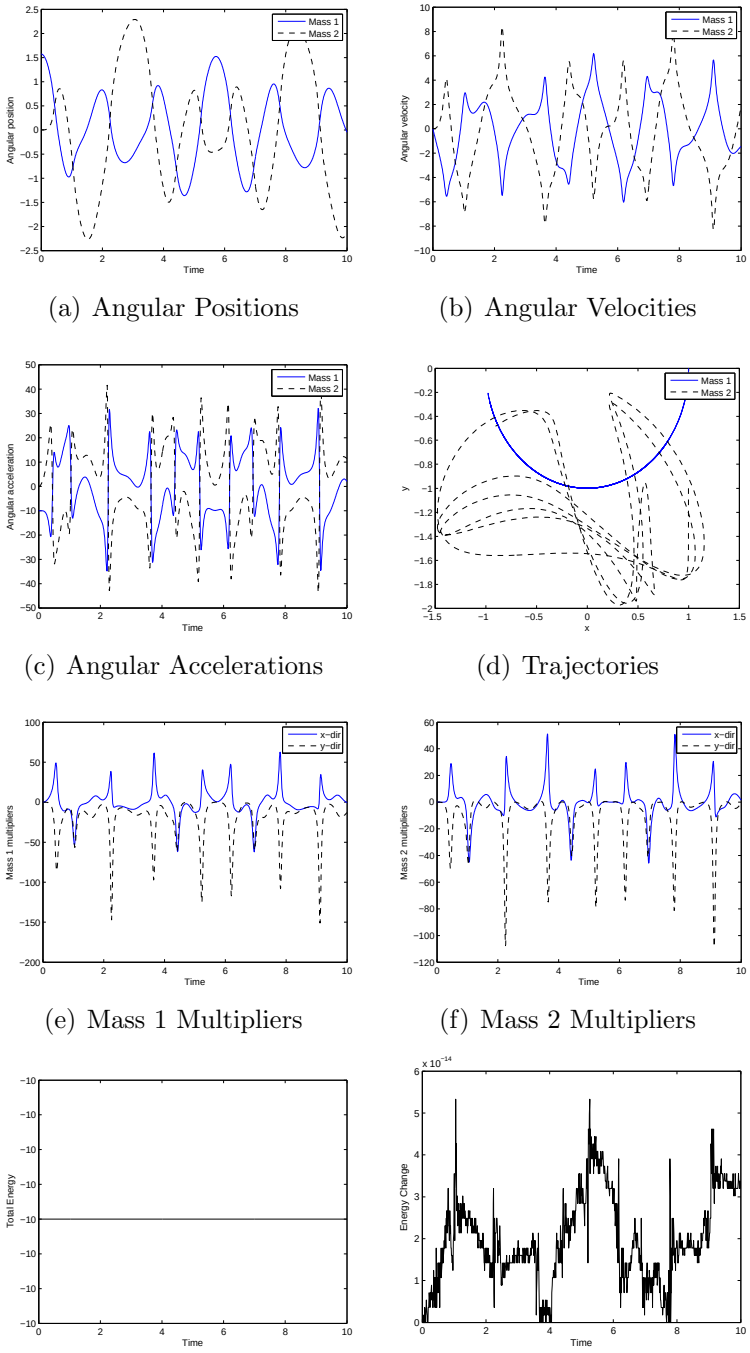


(g) Total Energy



(h) Energy Change

Figure 4.9: Double Pendulum - Option 3 - V0(1,1,1)



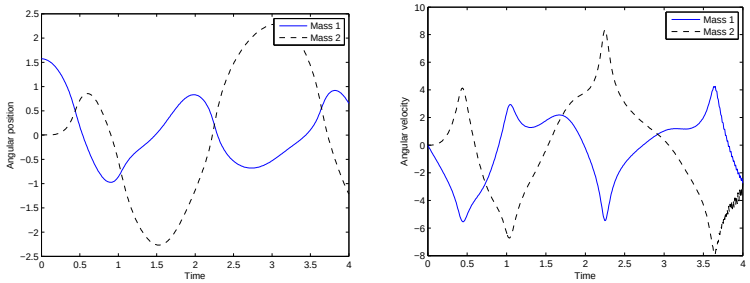
(a) Angular Positions (b) Angular Velocities

(c) Angular Accelerations (d) Trajectories

(e) Mass 1 Multipliers (f) Mass 2 Multipliers

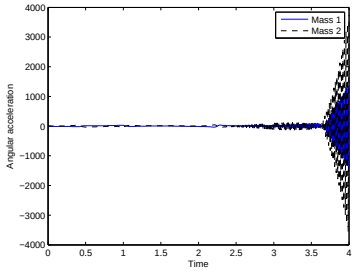
(g) Total Energy (h) Energy Change

Figure 4.10: Double Pendulum - Option 3 - V0(1,1,0)

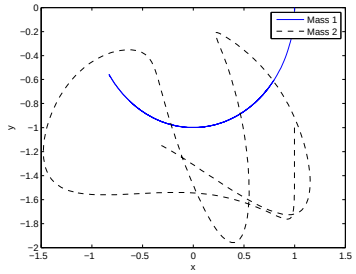


(a) Angular Positions

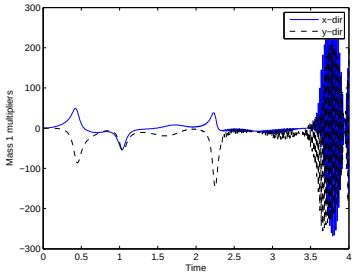
(b) Angular Velocities



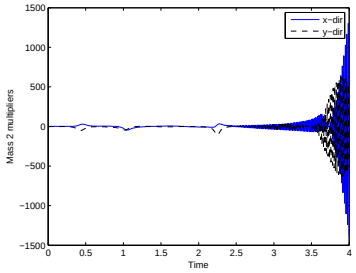
(c) Angular Accelerations



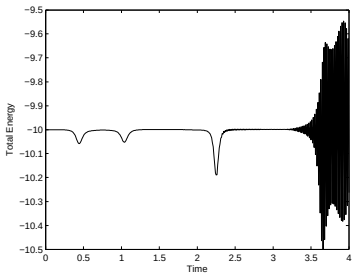
(d) Trajectories



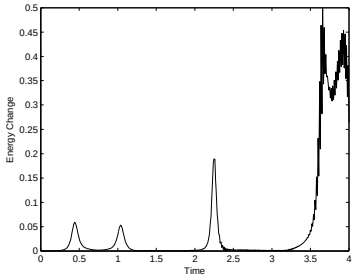
(e) Mass 1 Multipliers



(f) Mass 2 Multipliers

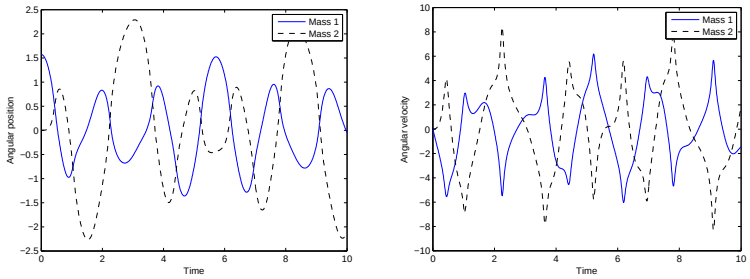


(g) Total Energy



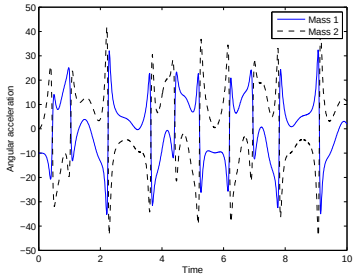
(h) Energy Change

Figure 4.11: Double Pendulum - Option 4 - U0(1,1,0)

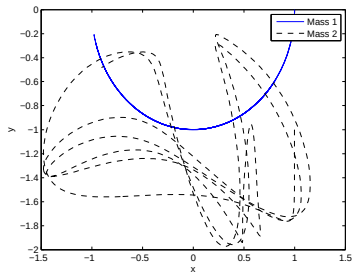


(a) Angular Positions

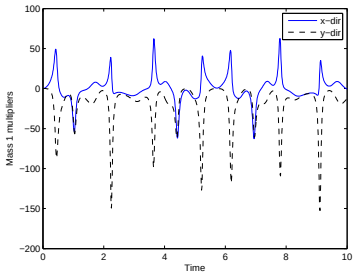
(b) Angular Velocities



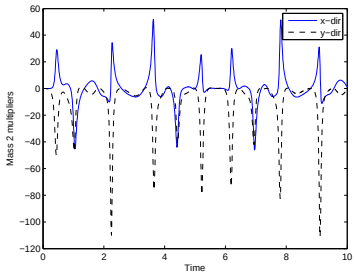
(c) Angular Accelerations



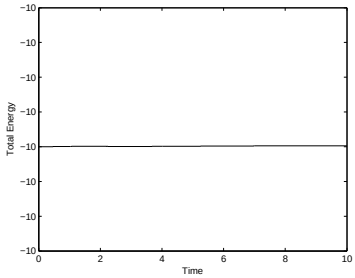
(d) Trajectories



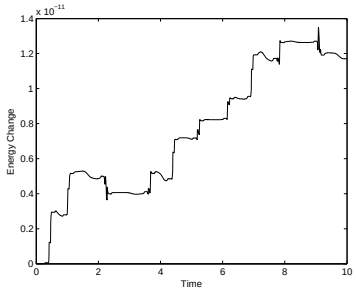
(e) Mass 1 Multipliers



(f) Mass 2 Multipliers

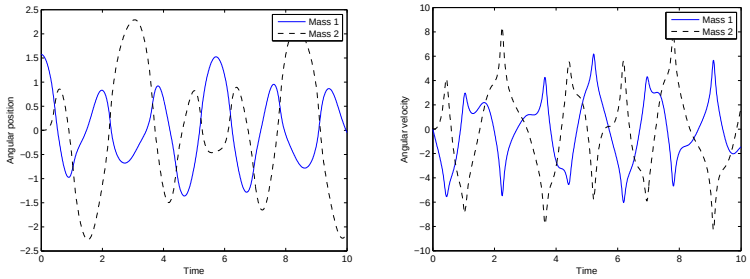


(g) Total Energy



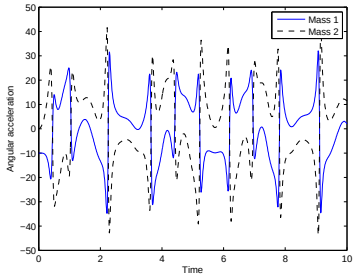
(h) Energy Change

Figure 4.12: Double Pendulum - Option 4 - V0(1,1,1)

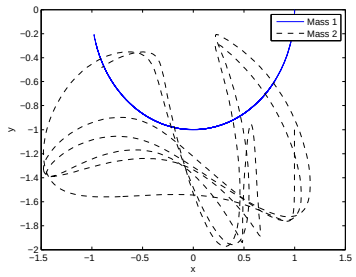


(a) Angular Positions

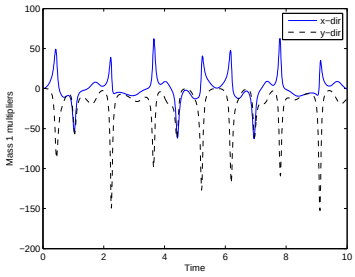
(b) Angular Velocities



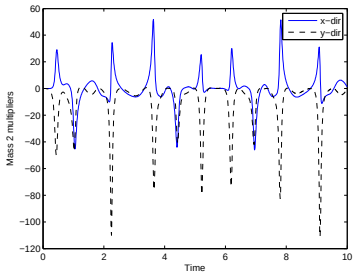
(c) Angular Accelerations



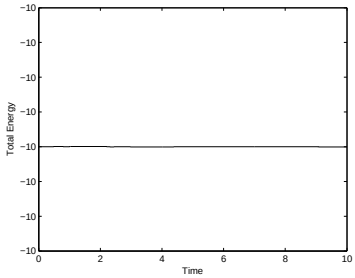
(d) Trajectories



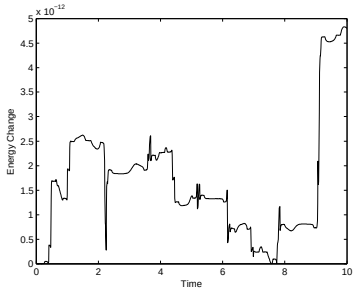
(e) Mass 1 Multipliers



(f) Mass 2 Multipliers



(g) Total Energy



(h) Energy Change

Figure 4.13: Double Pendulum - Option 4 - V0(1,1,0)

Table 4.1: Comparison of failed double pendulum with $\Delta t = 0.01$

Algorithm	Option	Endtime	CPU Time	Iterations	Iterations/Δt
U0(1,1,0)	1	4	0.444496	1375	3.4375
U0/V0(1,1,1)	1	4	2.207223	12174	30.435
V0(1,1,0)	1	4	2.195528	12173	30.4325
U0(1,1,0)	2	4	0.422997	1375	3.4375
U0(1,1,0)	3	4	0.40868	1375	3.4375
U0(1,1,0)	4	4	0.385041	1375	3.4375

Table 4.2: Comparison of successful double pendulum with $\Delta t = 0.01$

Algorithm	Option	Endtime	CPU Time	Iterations	Iterations/Δt
U0/V0(1,1,1)	2	10	0.831469	2999	2.999
V0(1,1,0)	2	10	0.837103	2998	2.998
U0/V0(1,1,1)	3	10	1.226276	5340	5.34
V0(1,1,0)	3	10	1.221351	5339	5.339
U0/V0(1,1,1)	4	10	4.634387	27197	27.197
V0(1,1,0)	4	10	4.612226	27197	27.197

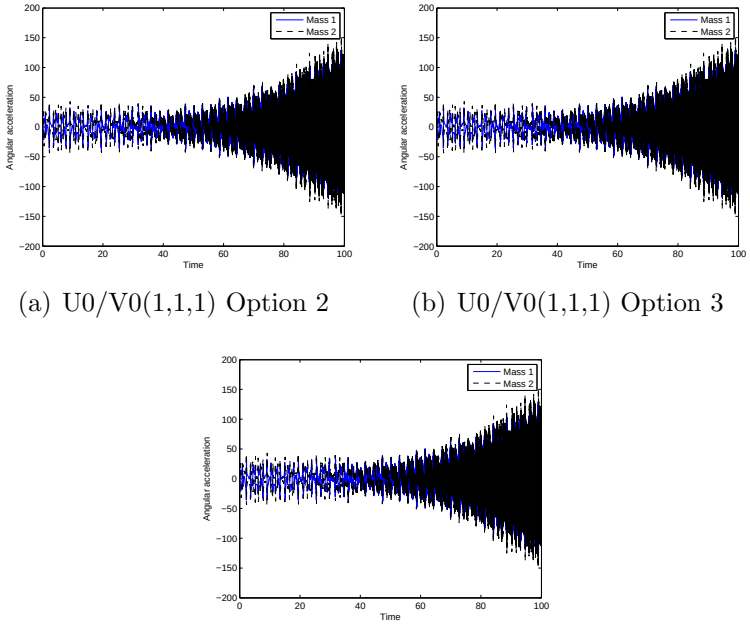
Discussion

Tables 4.1 and 4.2 give computational runtime information for the algorithms which failed (in nonlinear iterations) and those which were able to provide a full 10 second simulation, respectively. It can be seen that there are two fundamental sources of instability in the failed simulations: the accelerations and the Lagrange multipliers. These values are seen to oscillate with increasing magnitude until the nonlinear iterations are no longer able to converge. Notice that though the acceleration and Lagrange multipliers oscillate violently the velocities and displacements are smooth and match exactly the results of the stable solutions. To further test the 6 schemes which were able to provide a stable solution to 10 seconds, the simulation time was extended further to 100 seconds to ensure oscillations (if present) had sufficient time to manifest noticeably. Fig. 4.14 shows for long duration simulation the U0/V0(1,1,1) (midpoint rule with endpoint acceleration) algorithm

becomes unstable in angular accelerations for the remaining options 2, 3, and 4 leading to eventual failure of the iterative nonlinear solver.

Using option 4 the V0(1,1,0) algorithm is stable in accelerations to 100 seconds, but begins to show oscillatory behavior in the Lagrange multipliers and requires drastically more nonlinear iterations to converge than options 2 and 3. The remaining candidates for a stable long duration algorithm are then the V0(1,1,0) algorithm using options 2, and 3. The 100 second simulation results are shown in Figs. 4.15 and 4.16. Table 4.3 again shows CPU time and iterations for the 100 second simulation. Though both options 2 and 3 provide stable results, of note is the ability of option 2 to conserve energy almost to machine precision, as well as requiring nearly *half* the total number of iterations of option 3. This is an extremely noteworthy result, as this problem was used in [3] as demonstration of the necessity of controllable numerical dissipation (though as noted only option 1 and U0(1,1,0) were investigated there).

To demonstrate that the constraint equations at the position, velocity, and acceleration level are perfectly stable, the downfall of many index 2 DAE algorithms, the V0(1,1,0) algorithm was run for 500 seconds. Unlike the position level constraint, being an index 3 methodology the velocity and acceleration level constraint are not directly imposed by the algorithm. In reduced index methods (index 2 or index 1), the corresponding velocity and acceleration level constraints are those which are imposed directly by the algorithm. The problem of constraint drift plagues such algorithms as the position level constraint will drift from exact satisfaction unbounded. As discussed earlier, a wide variety of techniques have been developed to attempt to overcome this drift, but always come at the cost of reformulation of the natural system, and almost always with additional computational cost. Fig. 4.17 demonstrates that using the V0(1,1,0) algorithm in conjunction with option 2 the constraint at all 3 levels is perfectly stable.



(a) U0/V0(1,1,1) Option 2 (b) U0/V0(1,1,1) Option 3

(c) U0/V0(1,1,1) Option 4

Figure 4.14: Double pendulum simulation to 100 seconds

Convergence

According to the previously developed equation of motion time level theory, if each quantity in the equation are satisfied at a consistent time level the second order accuracy of the algorithm should be maintained. To ensure the methods under investigation indeed follow this theory convergence plots were made for the same 12 combinations of the base algorithms and options.

To create the convergence plots the problem was run to an end time of 0.25 seconds with the same parameters and tolerances as above. The "exact" solution was obtained using $\Delta t_{exact} = 2^{-10}$. Four additional numerical solutions were run in order to calculate error using $\Delta t_1 = 2^{-8}$, $\Delta t_2 = 2^{-7}$, $\Delta t_3 = 2^{-6}$, and $\Delta t_4 = 2^{-5}$. Figs. 4.18 and 4.19 confirm that, as expected, all 3 algorithms are exactly second order accurate regardless of the option under consideration.

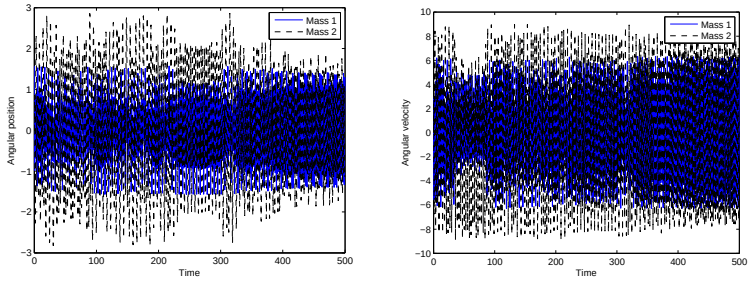
The Newmark method and Midpoint rule with endpoint acceleration both

Table 4.3: Long duration (100 sec) double pendulum with $\Delta t = 0.01$

Algorithm	Option	Endtime	CPU Time	Iterations	Iterations/Δt
U0/V0(1,1,1)	2	Unstable in accelerations			
V0(1,1,0)	2	100	15.87	29995	2.999
U0/V0(1,1,1)	3	Unstable in accelerations			
V0(1,1,0)	3	100	19.94	52152	5.215
U0/V0(1,1,1)	4	Unstable in accelerations			
V0(1,1,0)	4	100	56.07	287766	28.777

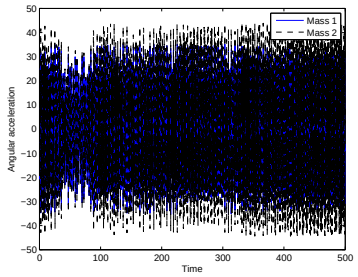
return values of $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{a}$ at time level t_{n+1} so no alignment is required for these algorithms. The Midpoint rule with midpoint acceleration, as the name suggests, returns values of acceleration at time level $t_{n+\frac{1}{2}}$. To consistently obtain values of acceleration at time 0.25 seconds the acceleration obtained in the final time step is extrapolated by half a time step as $\mathbf{a}_{end} = (1 + \frac{1}{2})\mathbf{a}_{n+1} - \frac{1}{2}\mathbf{a}_n$.

Similarly, to obtain consistent values of the Lagrange multipliers care must be taken in alignment while using option 2 and option 3. In these two options, the time level of the Lagrange multipliers output by the algorithm is $t_{n+\frac{1}{2}}$. Much like the accelerations, the values of the Lagrange multipliers obtained from the final time step are extrapolated by half a time step to obtain consistent values from each simulation (values at exactly $t = 0.25$). This extrapolation is $\boldsymbol{\lambda}_{end} = (1 + \frac{1}{2})\boldsymbol{\lambda}_{n+1} - \frac{1}{2}\boldsymbol{\lambda}_n$.

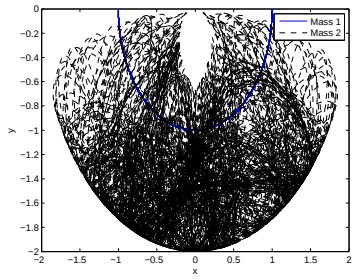


(a) Angular Positions

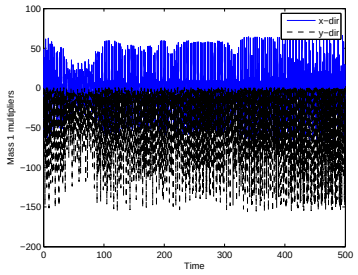
(b) Angular Velocities



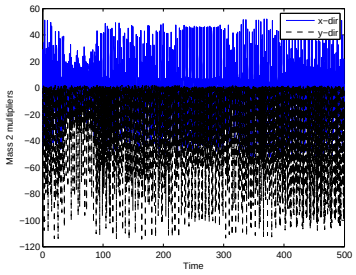
(c) Angular Accelerations



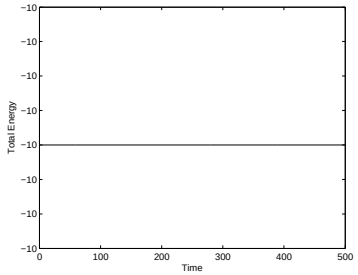
(d) Trajectories



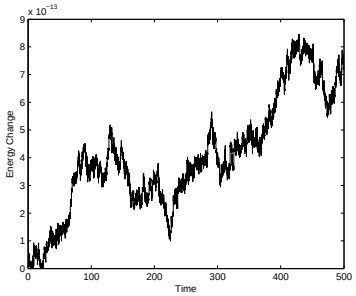
(e) Mass 1 Multipliers



(f) Mass 2 Multipliers

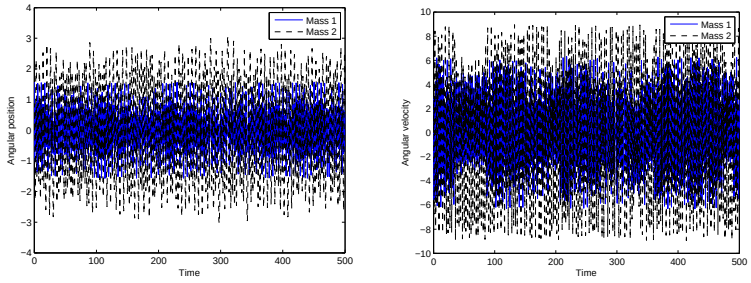


(g) Total Energy



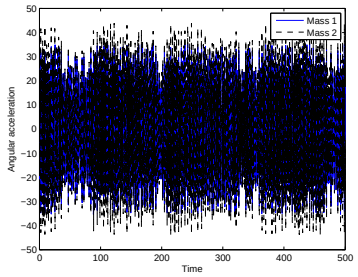
(h) Energy Change

Figure 4.15: Long Duration Double Pendulum - Option 2 - V0(1,1,0)

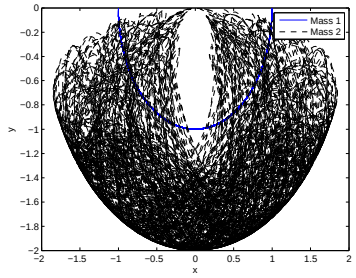


(a) Angular Positions

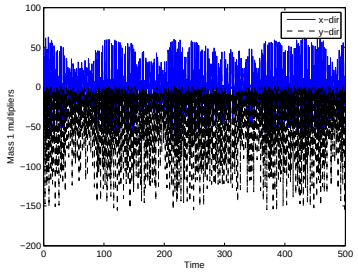
(b) Angular Velocities



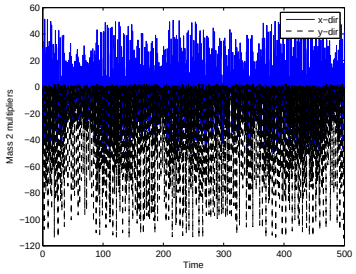
(c) Angular Accelerations



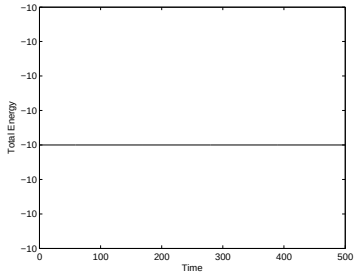
(d) Trajectories



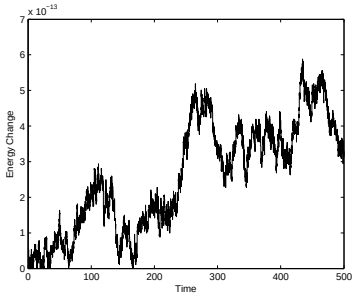
(e) Mass 1 Multipliers



(f) Mass 2 Multipliers



(g) Total Energy



(h) Energy Change

Figure 4.16: Long Duration Double Pendulum - Option 3 - V0(1,1,0)

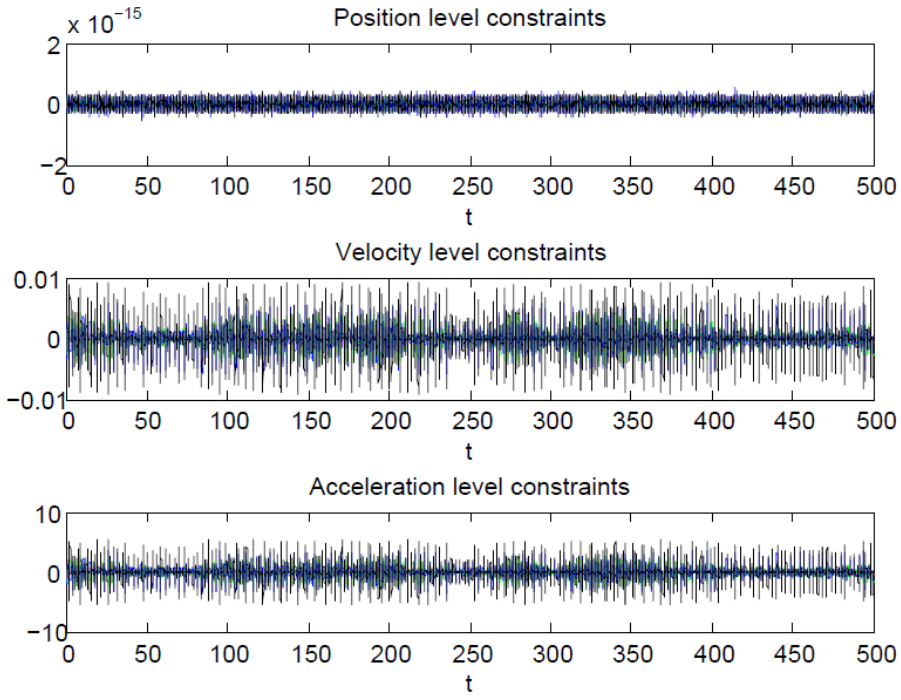
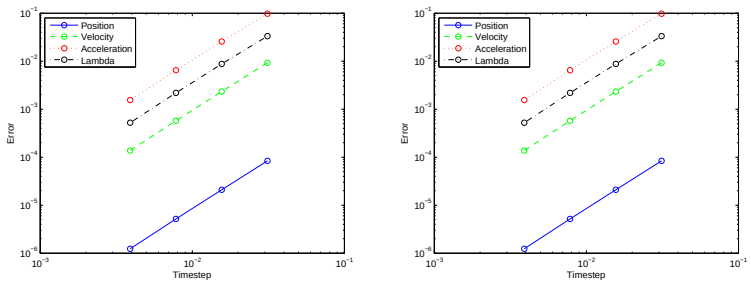
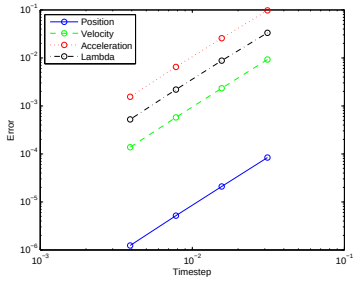


Figure 4.17: Constraint Satisfaction to 500 sec., V0(1,1,0) Option 2

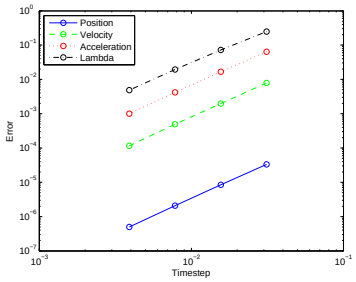


(a) Option 1 U0(1,1,0)

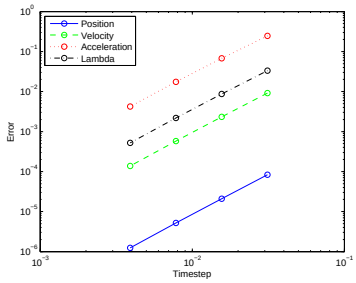
(b) Option 2 U0(1,1,0)



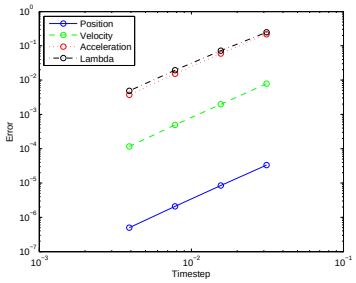
(c) Option 1 U0/V0(1,1,1)



(d) Option 2 U0/V0(1,1,1)

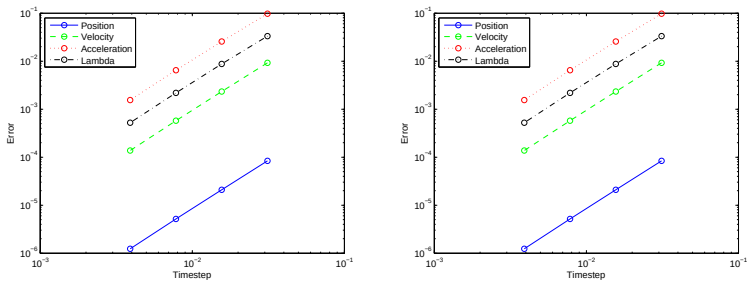


(e) Option 1 V0(1,1,0)



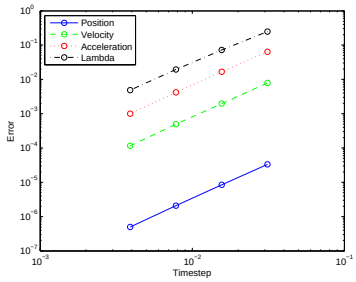
(f) Option 2 V0(1,1,0)

Figure 4.18: Double Pendulum - Option 1, 2 Convergence

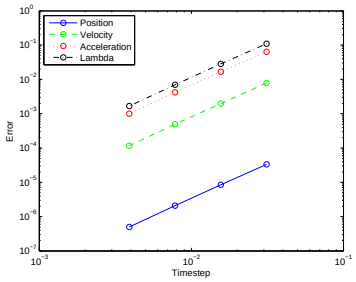


(a) Option 3 U0(1,1,0)

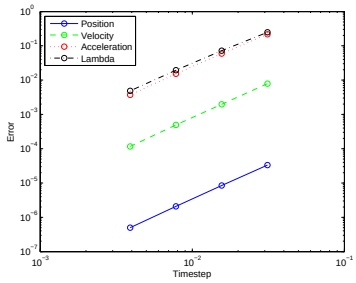
(b) Option 4 U0(1,1,0)



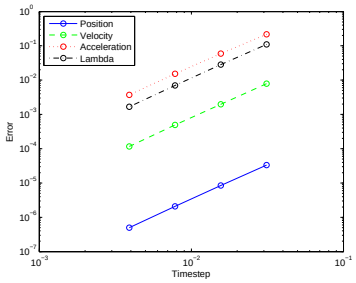
(c) Option 3 U0/V0(1,1,1)



(d) Option 4 U0/V0(1,1,1)



(e) Option 3 V0(1,1,0)



(f) Option 4 V0(1,1,0)

Figure 4.19: Double Pendulum - Option 3, 4 Convergence

Two Field Form Results

The two field algorithm in its v-form representation (equivalent to V0(1,1,0) algorithm) results are shown in Figs. 4.20-4.23. All simulation parameters are identical to the single field form simulations above. The jacobian matrices in the v-form representation of the two field form under options 1, 2, 3, and 4 are:

Option 1:

$$\mathbf{J} = \begin{bmatrix} \frac{1}{\Delta t}\mathbf{M} + \frac{\Delta t}{4}\mathbf{J}_{B\lambda}(\mathbf{u}_{n+1}, \boldsymbol{\lambda}_{n+1}) & \frac{1}{2}\mathbf{B}(\mathbf{u}_{n+1})^T \\ \mathbf{B}(\mathbf{u}_{n+1}) & \mathbf{0} \end{bmatrix} \tag{4.20}$$

Option 2:

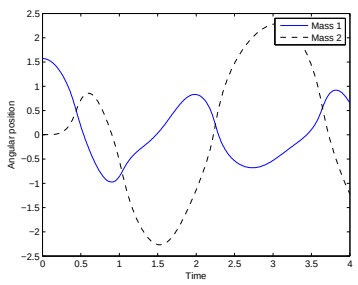
$$\mathbf{J} = \begin{bmatrix} \frac{1}{\Delta t}\mathbf{M} + \frac{\Delta t}{4}\mathbf{J}_{B\lambda}(\widetilde{\mathbf{u}}, \boldsymbol{\lambda}_{n+1}) & \mathbf{B}(\widetilde{\mathbf{u}})^T \\ \mathbf{B}(\mathbf{u}_{n+1}) & \mathbf{0} \end{bmatrix} \tag{4.21}$$

Option 3:

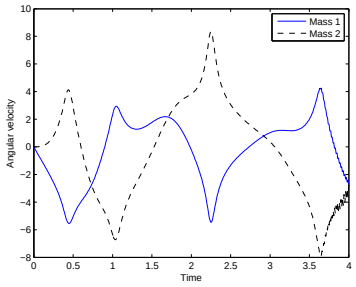
$$\mathbf{J} = \begin{bmatrix} \frac{1}{\Delta t}\mathbf{M} + \frac{\Delta t}{4}\mathbf{J}_{B\lambda}(\mathbf{u}_{n+1}, \boldsymbol{\lambda}_{n+1}) & \frac{1}{2}(\mathbf{B}(\mathbf{u}_n) + \mathbf{B}(\mathbf{u}_{n+1}))^T \\ \mathbf{B}(\mathbf{u}_{n+1}) & \mathbf{0} \end{bmatrix} \tag{4.22}$$

Option 4:

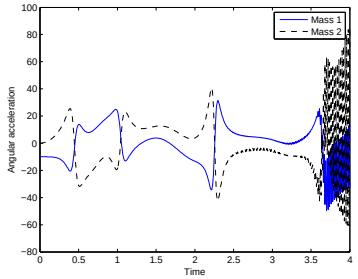
$$\mathbf{J} = \begin{bmatrix} \frac{1}{\Delta t}\mathbf{M} + \frac{\Delta t}{8}\mathbf{J}_{B\lambda}(\widetilde{\mathbf{u}}, \boldsymbol{\lambda}_{n+1}) & \frac{1}{2}\mathbf{B}(\widetilde{\mathbf{u}})^T \\ \mathbf{B}(\mathbf{u}_{n+1}) & \mathbf{0} \end{bmatrix} \tag{4.23}$$



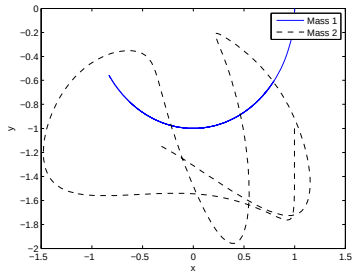
(a) Angular Positions



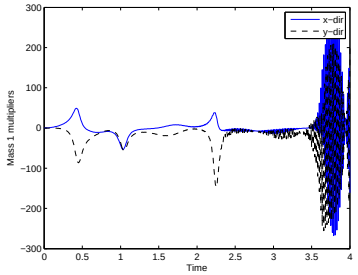
(b) Angular Velocities



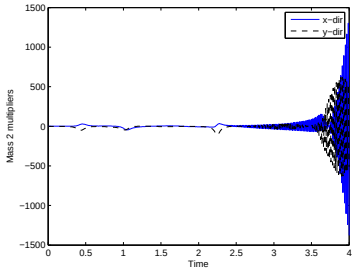
(c) Angular Accelerations



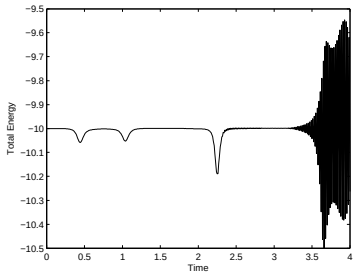
(d) Trajectories



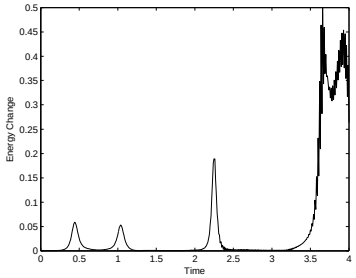
(e) Mass 1 Multipliers



(f) Mass 2 Multipliers

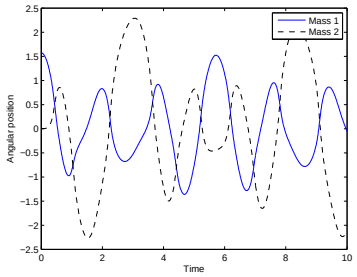


(g) Total Energy

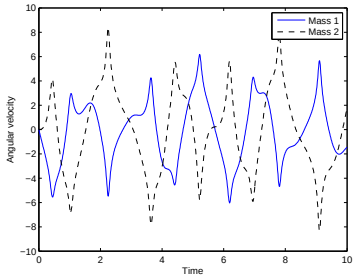


(h) Energy Change

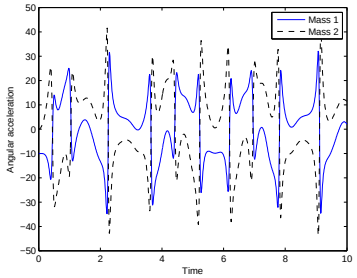
Figure 4.20: Double Pendulum - Option 1 - Two Field Form



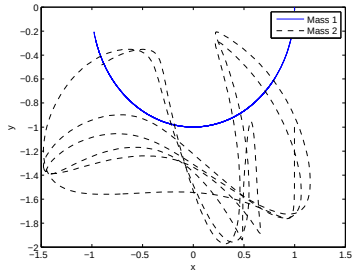
(a) Angular Positions



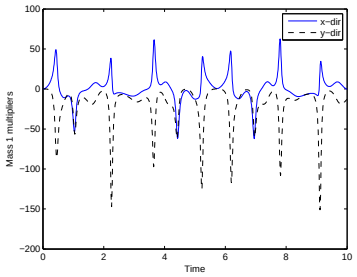
(b) Angular Velocities



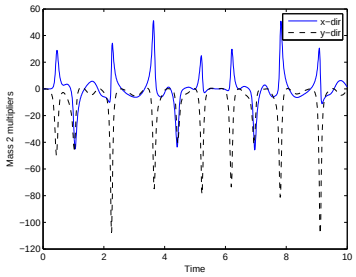
(c) Angular Accelerations



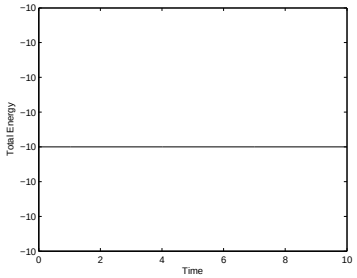
(d) Trajectories



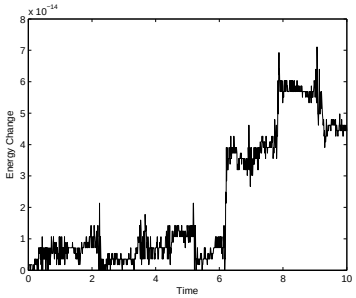
(e) Mass 1 Multipliers



(f) Mass 2 Multipliers

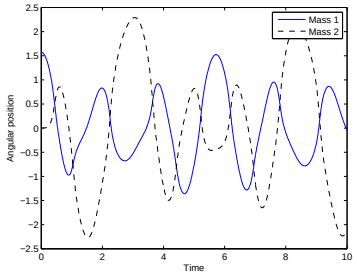


(g) Total Energy

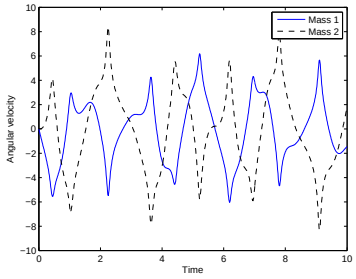


(h) Energy Change

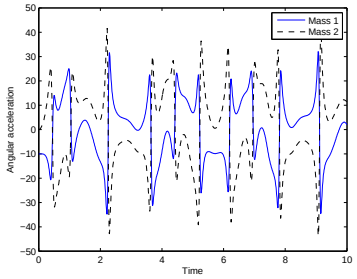
Figure 4.21: Double Pendulum - Option 2 - Two Field Form



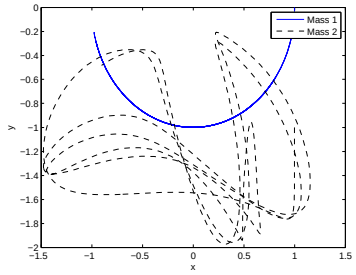
(a) Angular Positions



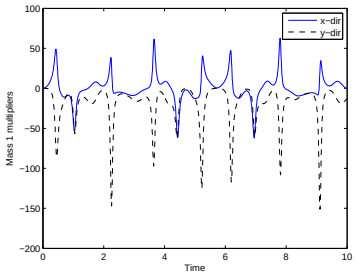
(b) Angular Velocities



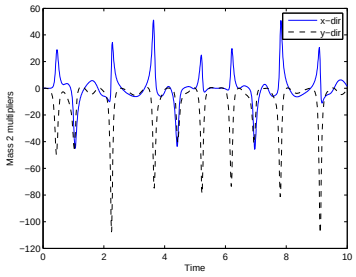
(c) Angular Accelerations



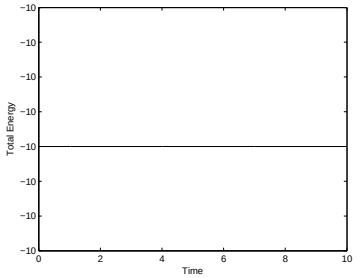
(d) Trajectories



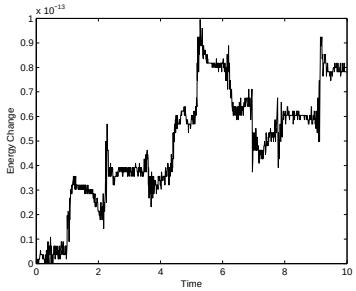
(e) Mass 1 Multipliers



(f) Mass 2 Multipliers

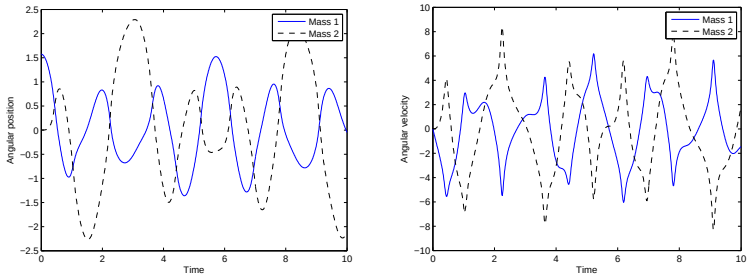


(g) Total Energy



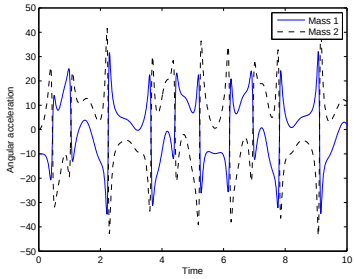
(h) Energy Change

Figure 4.22: Double Pendulum - Option 3 - Two Field Form

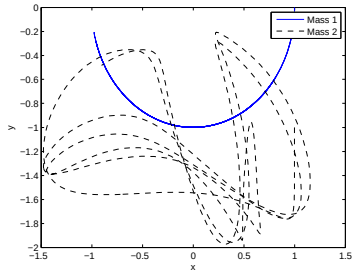


(a) Angular Positions

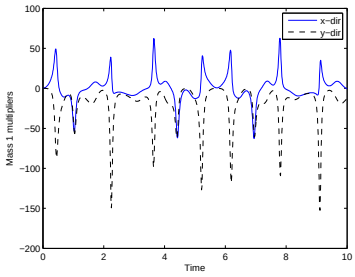
(b) Angular Velocities



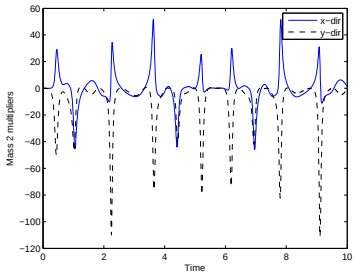
(c) Angular Accelerations



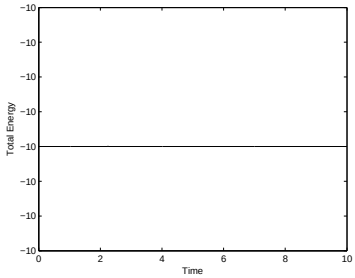
(d) Trajectories



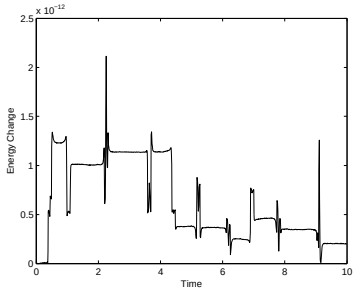
(e) Mass 1 Multipliers



(f) Mass 2 Multipliers



(g) Total Energy



(h) Energy Change

Figure 4.23: Double Pendulum - Option 4 - Two Field Form

4.2 Flexible Multibody Dynamics

Next these methods were tested on systems which include both rigid and flexible elements, the most practical and general type of system in real-world applications. The same solution space of 12 total algorithms was investigated for two different models. The first is a system of four masses connected by two rigid bars and two flexible truss elements, which has been investigated by a variety of authors. This provides a common ground for comparison not only of the 12 algorithms described here, but also against existing methods found in the literature. The second example adds another level of complexity by analyzing a system of thin beams whose potential is no longer simply a quadratic function of the displacements. As such, it demonstrates the general nature the methods developed here for systems which require additional formulation for energy conservation.

4.2.1 Four Mass System

In this example a system of four interconnected masses floating in 3-dimensional space is considered. Discussed at length by Gonzalez in [42], the system involves two analytically rigid bars and two flexible truss elements. The truss elements are treated by the energy-momentum method described in chapter 2. That is, they are composed of a standard linear material model but are geometrically nonlinear (Green strain). The system model is shown in Fig. 4.24.

The flexible bars have stiffness $V_1 = 100$ and $V_2 = 1000$, both with natural spring length of 1 unit. The problem consists of 12 degrees of freedom and 2 Lagrange multipliers. That is, the unknowns are:

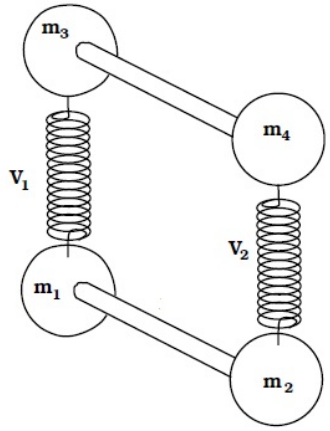


Figure 4.24: Four Mass System

$$\mathbf{u} = \begin{bmatrix} x_1 & y_1 & z_1 & x_2 & y_2 & z_2 & x_3 & y_3 & z_3 & x_4 & y_4 & z_4 \end{bmatrix}^T \tag{4.24}$$

$$\dot{\mathbf{u}} = \begin{bmatrix} \dot{x}_1 & \dot{y}_1 & \dot{z}_1 & \dot{x}_2 & \dot{y}_2 & \dot{z}_2 & \dot{x}_3 & \dot{y}_3 & \dot{z}_3 & \dot{x}_4 & \dot{y}_4 & \dot{z}_4 \end{bmatrix}^T \tag{4.25}$$

$$\ddot{\mathbf{u}} = \begin{bmatrix} \ddot{x}_1 & \ddot{y}_1 & \ddot{z}_1 & \ddot{x}_2 & \ddot{y}_2 & \ddot{z}_2 & \ddot{x}_3 & \ddot{y}_3 & \ddot{z}_3 & \ddot{x}_4 & \ddot{y}_4 & \ddot{z}_4 \end{bmatrix}^T \tag{4.26}$$

$$\boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 & \lambda_2 \end{bmatrix}^T \tag{4.27}$$

Two constraint equations serve to keep the length of both rigid bars constant ($L_{1-2} = L_{3-4} = 1$):

$$\boldsymbol{\Phi} = \begin{bmatrix} \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} - L_{1-2} \\ \sqrt{(x_4 - x_3)^2 + (y_4 - y_3)^2 + (z_4 - z_3)^2} - L_{3-4} \end{bmatrix} \tag{4.28}$$

The $\mathbf{B}$ matrix is defined as follows, where L_1 and L_2 is the distance separating nodes 1 and 2 and nodes 3 and 4 respectively (not the constant value L_{1-2} and L_{3-4} to which they are constrained).

$$B = \begin{bmatrix} B11 & B12 \\ B21 & B22 \end{bmatrix} \tag{4.29}$$

$$B11 = \begin{bmatrix} \frac{x_1-x_2}{L_1} & \frac{y_1-y_2}{L_1} & \frac{z_1-z_2}{L_1} & \frac{x_2-x_1}{L_1} & \frac{y_2-y_1}{L_1} & \frac{z_2-z_1}{L_1} \end{bmatrix} \tag{4.30}$$

$$B22 = \begin{bmatrix} \frac{x_3-x_4}{L_2} & \frac{y_3-y_4}{L_2} & \frac{z_3-z_4}{L_2} & \frac{x_4-x_3}{L_2} & \frac{y_4-y_3}{L_2} & \frac{z_4-z_3}{L_2} \end{bmatrix} \tag{4.31}$$

$$B12 = B21 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.32}$$

$$\tag{4.33}$$

The mass matrix is defined as:

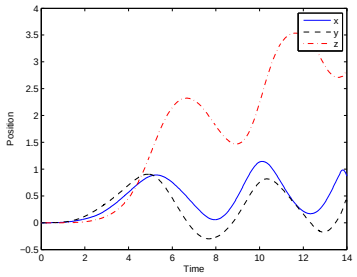
$$\mathbf{M} = diag(\begin{bmatrix} 1 & 1 & 1 & 3 & 3 & 3 & 2.3 & 2.3 & 2.3 & 1.7 & 1.7 & 1.7 \end{bmatrix}) \tag{4.34}$$

The initial conditions for this simulation were taken as:

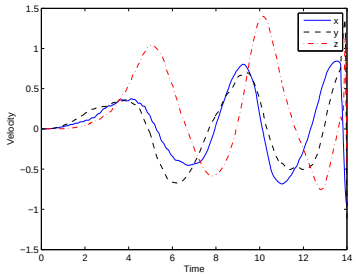
$$\mathbf{u_0} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}^T \tag{4.35}$$

$$\dot{\mathbf{u}}_\mathbf{0} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{2}{1.7} \end{bmatrix}^T \tag{4.36}$$

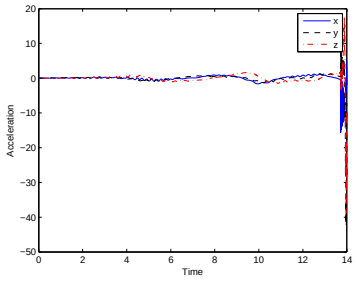
To again search the solution space of non-dissipative algorithms which the GSSSS family provides each algorithm was run using the same four options as in the previous rigid body example. The full response for each simulation are shown in Figs. 4.25-4.35, with the CPU time and iteration counts shown in Table 4.4. Each algorithm was again run using $\Delta t = 0.01$ and a nonlinear iteration tolerance of 10^{-8} using the a-form representation of the single field form algorithm. Discussion of the results follows the figures.



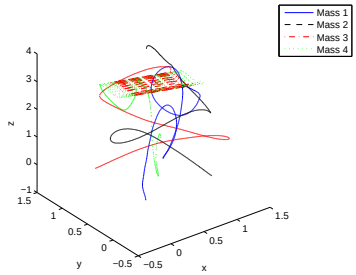
(a) Mass 1 Position



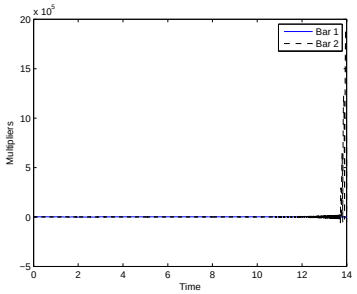
(b) Mass 1 Velocity



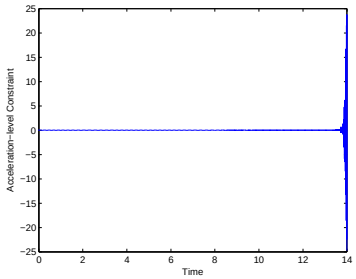
(c) Mass 1 Acceleration



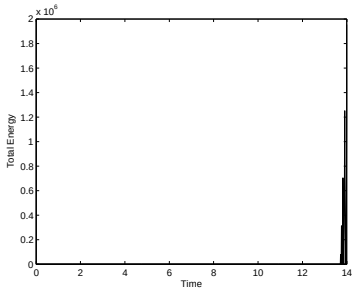
(d) Trajectories



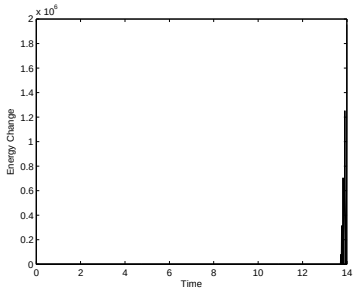
(e) Multipliers



(f) Acceleration Constraint

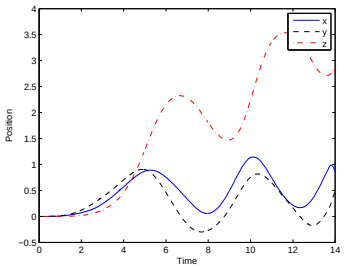


(g) Total Energy

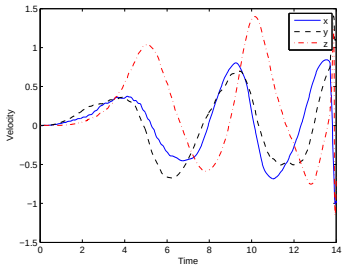


(h) Energy Change

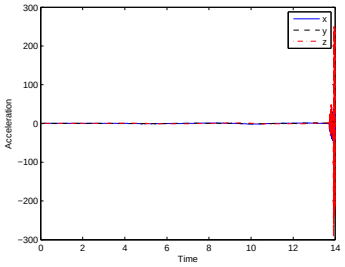
Figure 4.25: Four Mass System - Option 1 - U0(1,1,0)



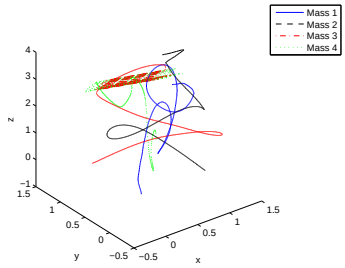
(a) Mass 1 Position



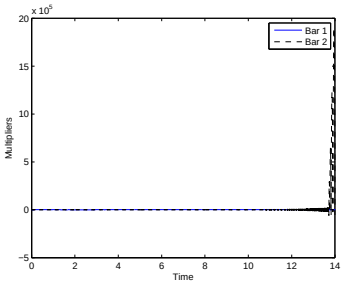
(b) Mass 1 Velocity



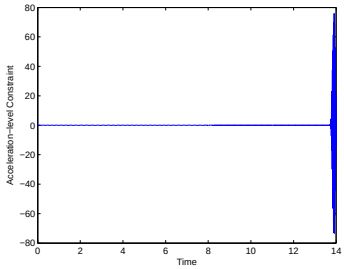
(c) Mass 1 Acceleration



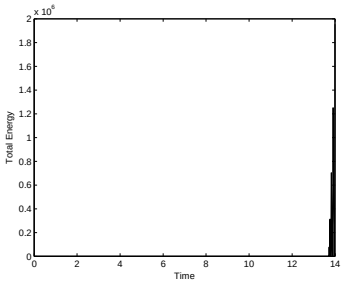
(d) Trajectories



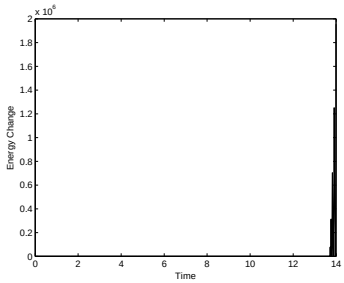
(e) Multipliers



(f) Acceleration Constraint

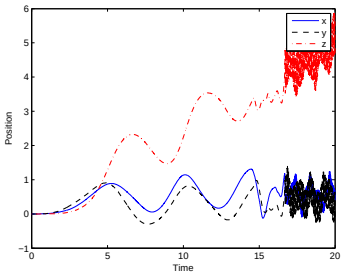


(g) Total Energy

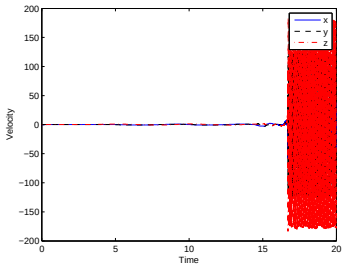


(h) Energy Change

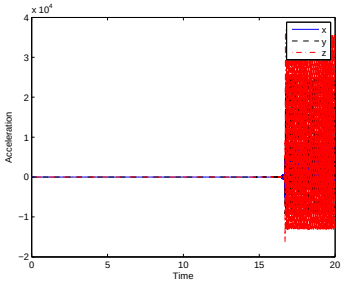
Figure 4.26: Four Mass System - Option 1 - V0(1,1,1)



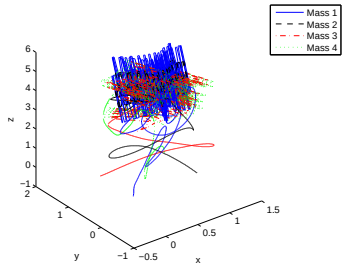
(a) Mass 1 Position



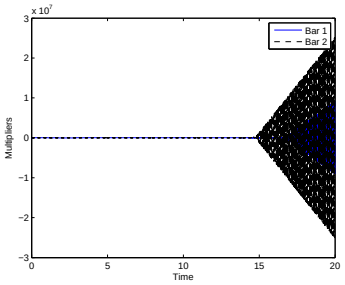
(b) Mass 1 Velocity



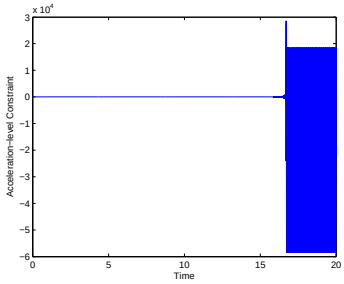
(c) Mass 1 Acceleration



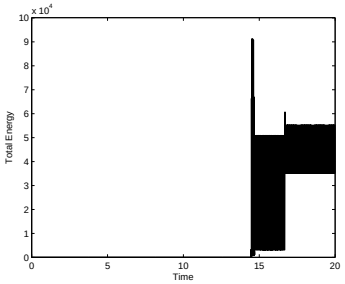
(d) Trajectories



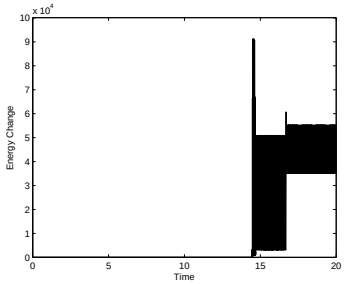
(e) Multipliers



(f) Acceleration Constraint

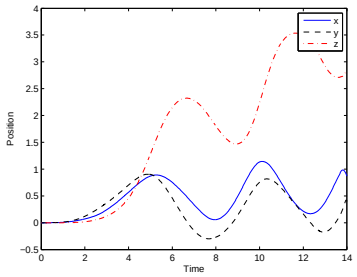


(g) Total Energy

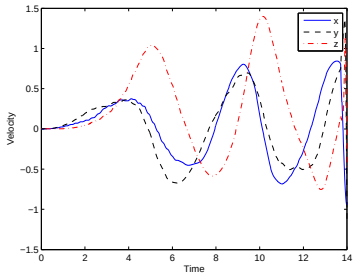


(h) Energy Change

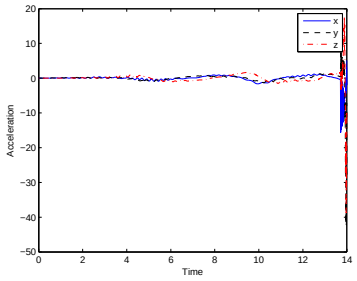
Figure 4.27: Four Mass System - Option 1 - V0(1,1,0)



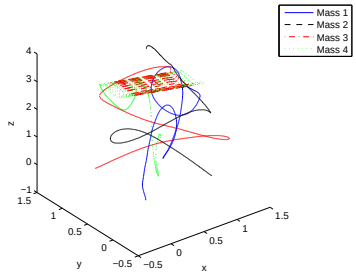
(a) Mass 1 Position



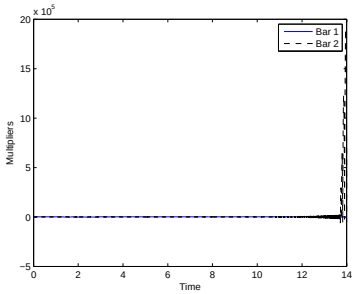
(b) Mass 1 Velocity



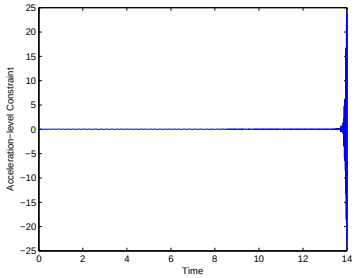
(c) Mass 1 Acceleration



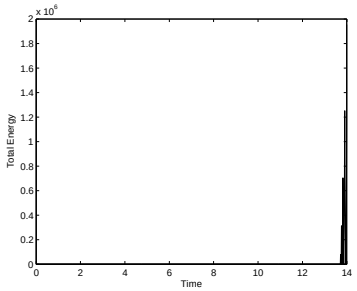
(d) Trajectories



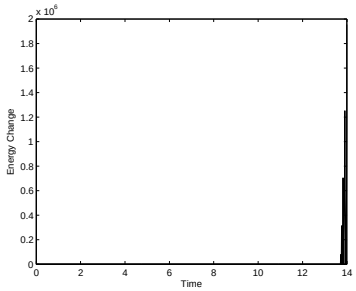
(e) Multipliers



(f) Acceleration Constraint

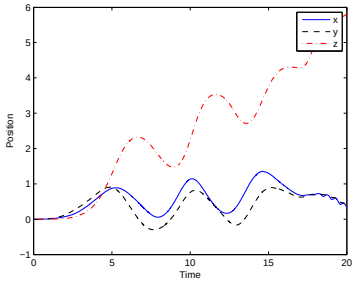


(g) Total Energy

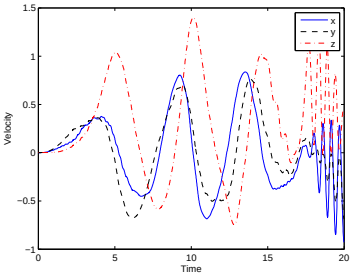


(h) Energy Change

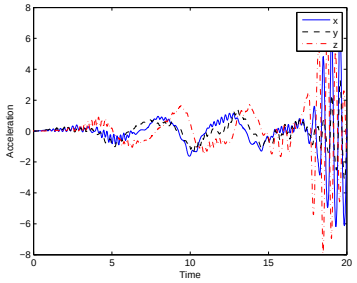
Figure 4.28: Four Mass System - Option 2 - U0(1,1,0)



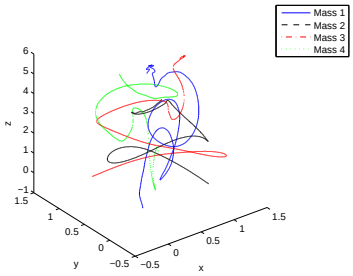
(a) Mass 1 Position



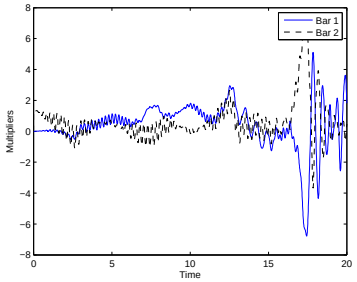
(b) Mass 1 Velocity



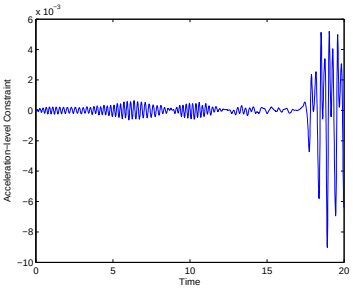
(c) Mass 1 Acceleration



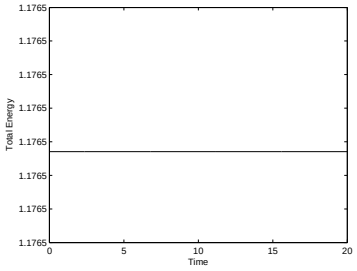
(d) Trajectories



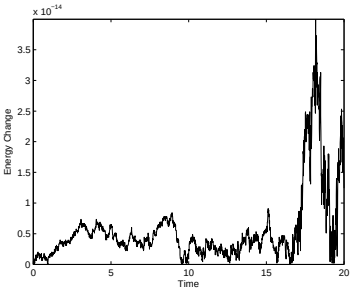
(e) Multipliers



(f) Acceleration Constraint

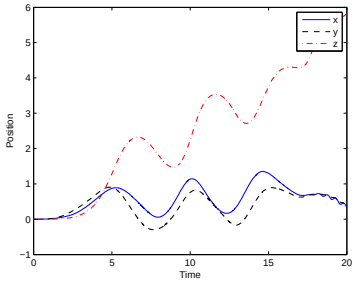


(g) Total Energy

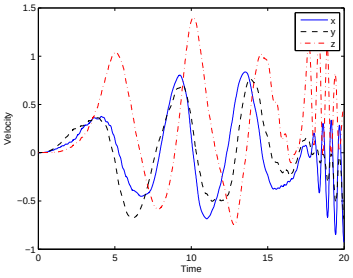


(h) Energy Change

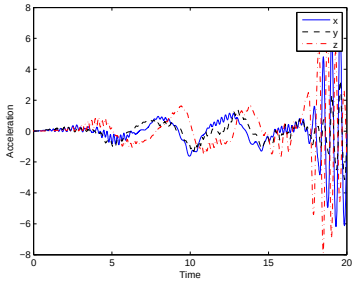
Figure 4.29: Four Mass System - Option 2 - V0(1,1,1)



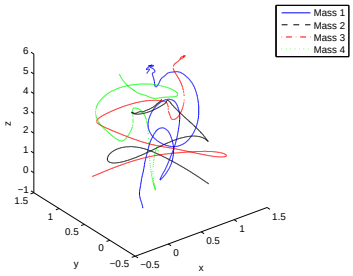
(a) Mass 1 Position



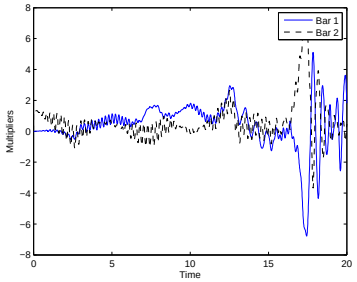
(b) Mass 1 Velocity



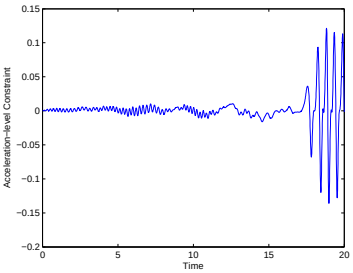
(c) Mass 1 Acceleration



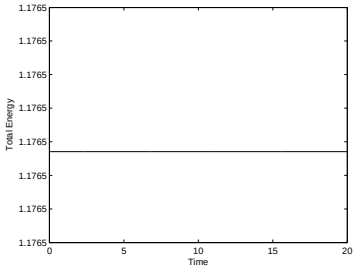
(d) Trajectories



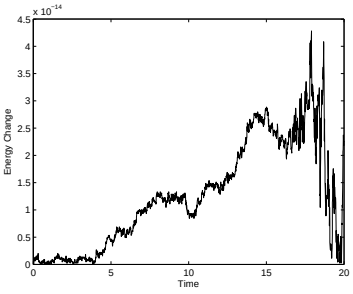
(e) Multipliers



(f) Acceleration Constraint

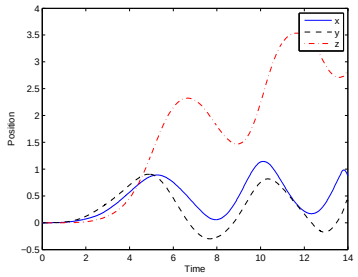


(g) Total Energy

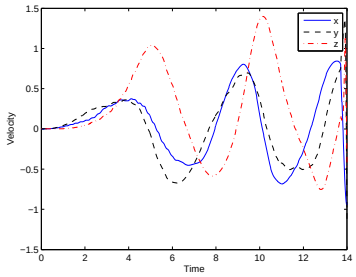


(h) Energy Change

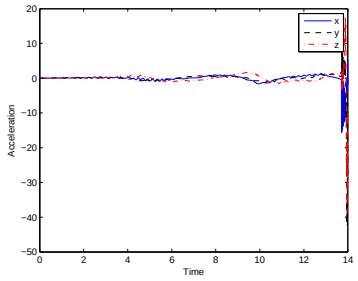
Figure 4.30: Four Mass System - Option 2 - V0(1,1,0)



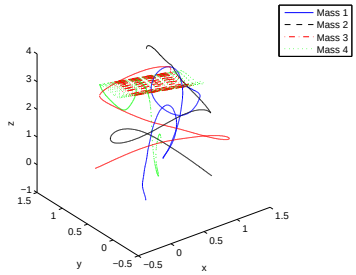
(a) Mass 1 Position



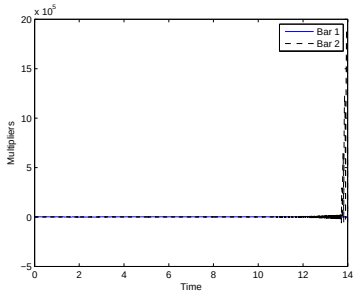
(b) Mass 1 Velocity



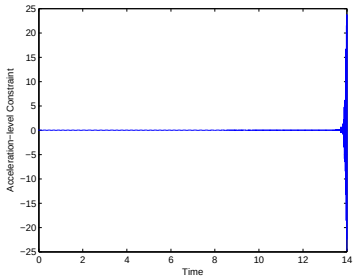
(c) Mass 1 Acceleration



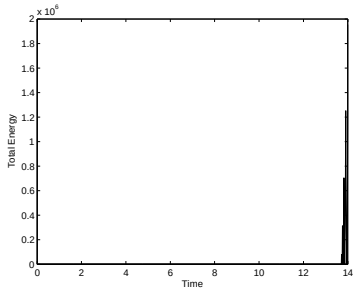
(d) Trajectories



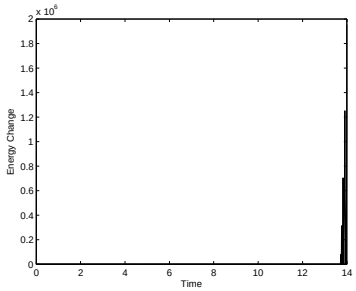
(e) Multipliers



(f) Acceleration Constraint

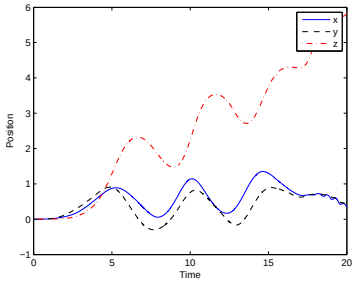


(g) Total Energy

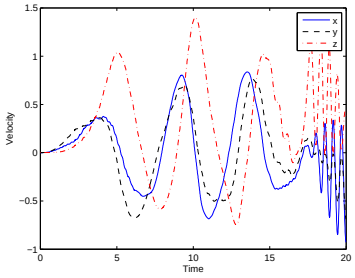


(h) Energy Change

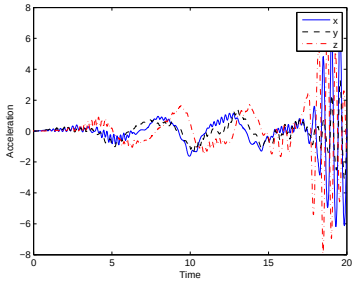
Figure 4.31: Four Mass System - Option 3 - U0(1,1,0)



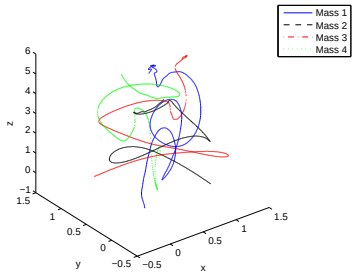
(a) Mass 1 Position



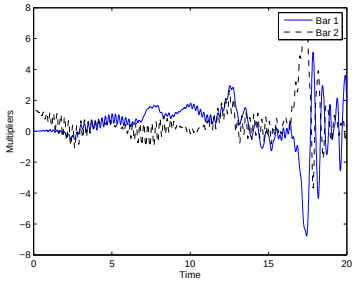
(b) Mass 1 Velocity



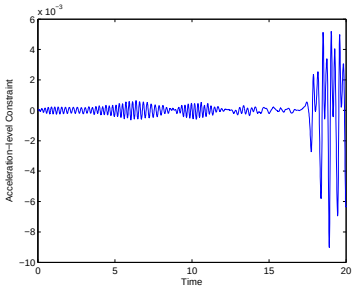
(c) Mass 1 Acceleration



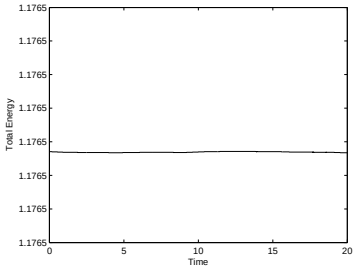
(d) Trajectories



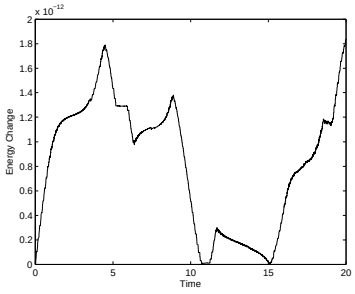
(e) Multipliers



(f) Acceleration Constraint

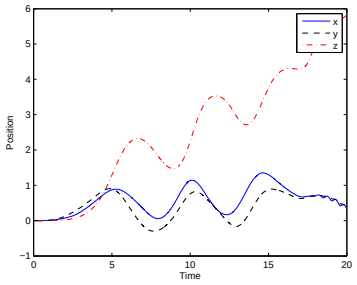


(g) Total Energy

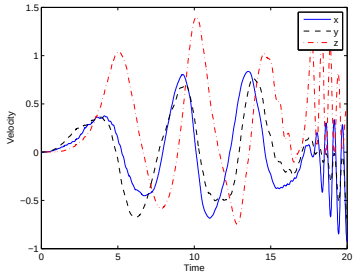


(h) Energy Change

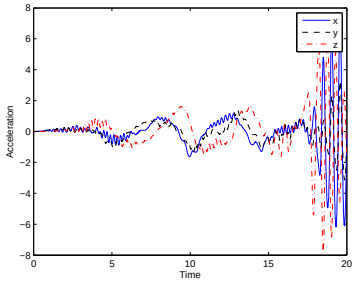
Figure 4.32: Four Mass System - Option 3 - V0(1,1,1)



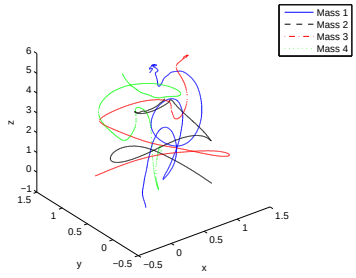
(a) Mass 1 Position



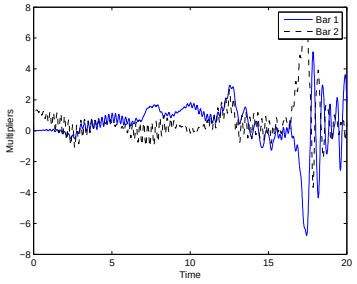
(b) Mass 1 Velocity



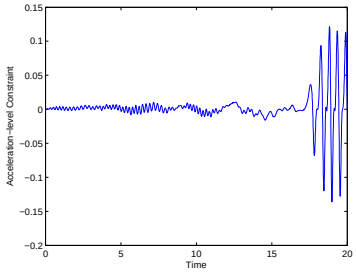
(c) Mass 1 Acceleration



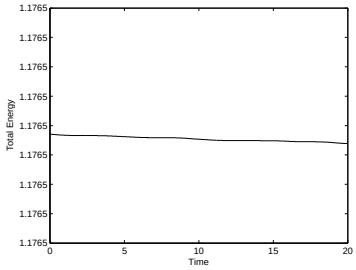
(d) Trajectories



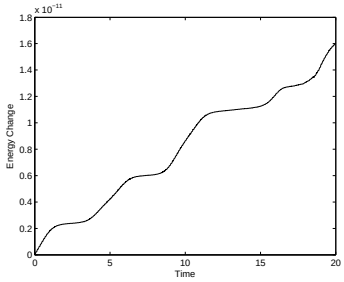
(e) Multipliers



(f) Acceleration Constraint

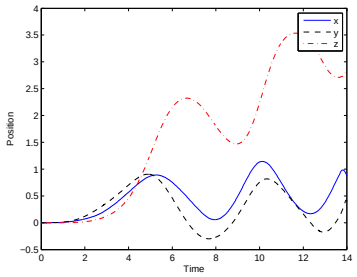


(g) Total Energy

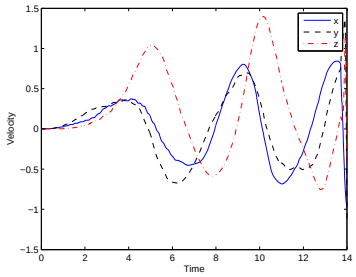


(h) Energy Change

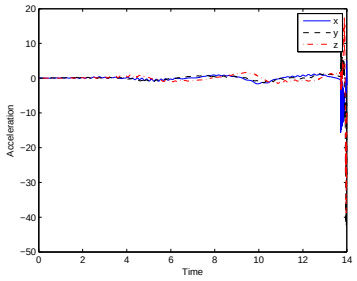
Figure 4.33: Four Mass System - Option 3 - V0(1,1,0)



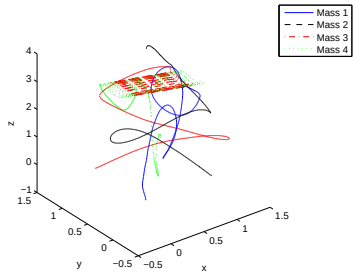
(a) Mass 1 Position



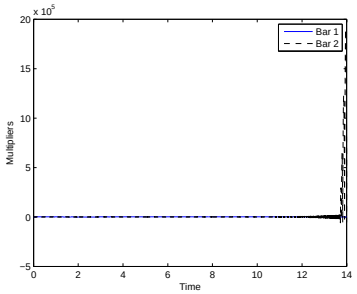
(b) Mass 1 Velocity



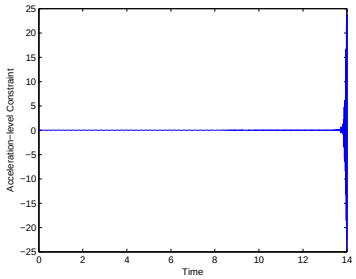
(c) Mass 1 Acceleration



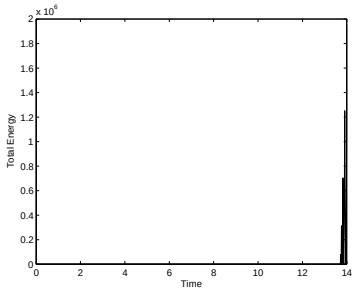
(d) Trajectories



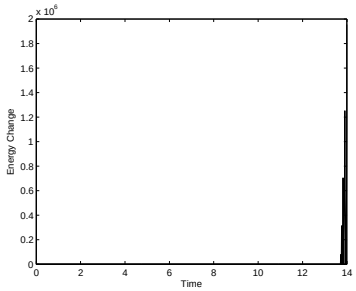
(e) Multipliers



(f) Acceleration Constraint

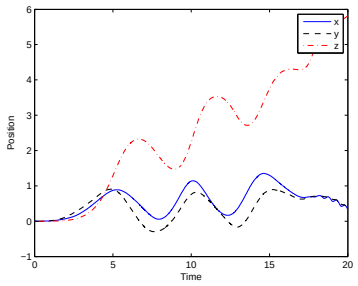


(g) Total Energy

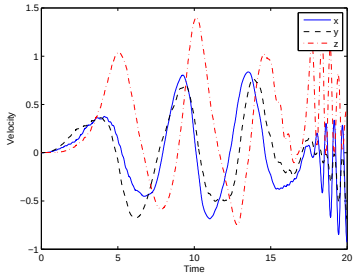


(h) Energy Change

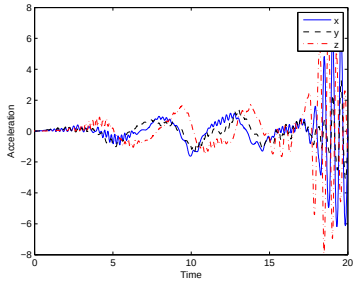
Figure 4.34: Four Mass System - Option 4 - U0(1,1,0)



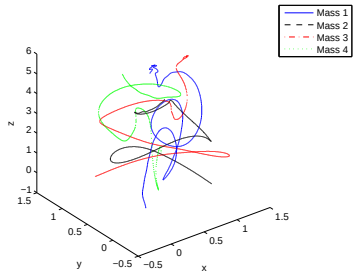
(a) Mass 1 Position



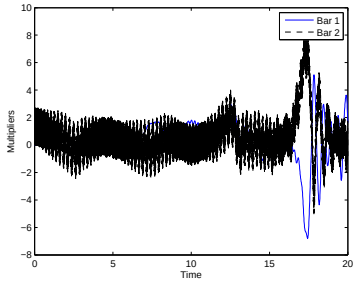
(b) Mass 1 Velocity



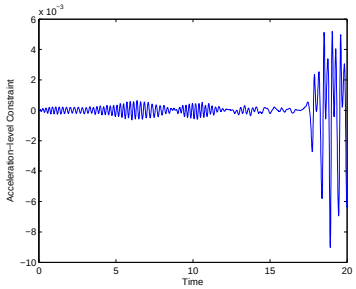
(c) Mass 1 Acceleration



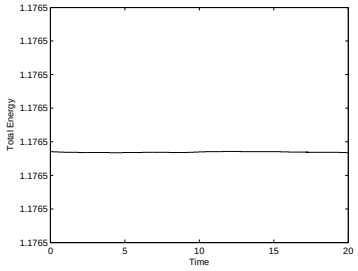
(d) Trajectories



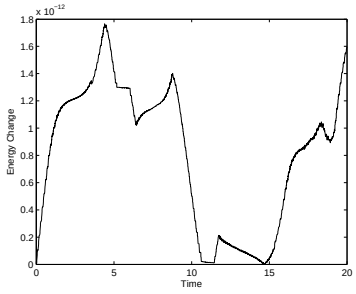
(e) Multipliers



(f) Acceleration Constraint

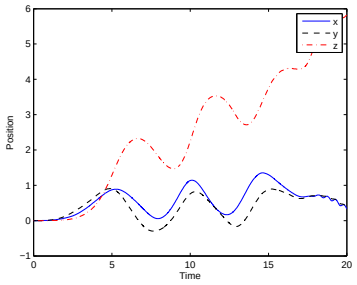


(g) Total Energy

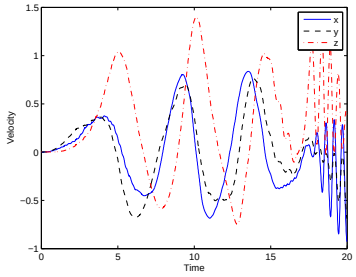


(h) Energy Change

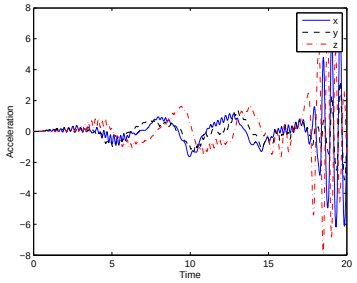
Figure 4.35: Four Mass System - Option 4 - V0(1,1,1)



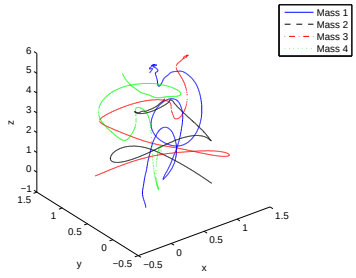
(a) Mass 1 Position



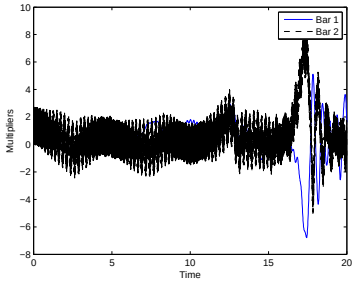
(b) Mass 1 Velocity



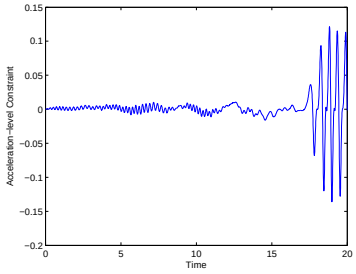
(c) Mass 1 Acceleration



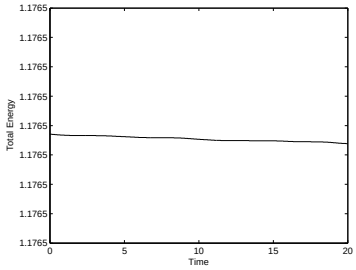
(d) Trajectories



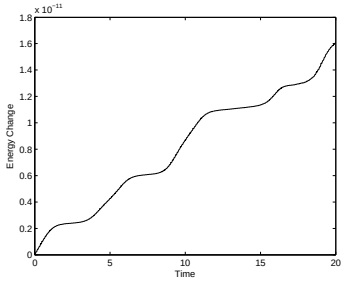
(e) Multipliers



(f) Acceleration Constraint



(g) Total Energy



(h) Energy Change

Figure 4.36: Four Mass System - Option 4 - V0(1,1,0)

Table 4.4: Four mass system with $\Delta t = 0.01$

Algorithm	Option	Endtime	CPU Time	Iterations	Iterations/Δt
U0(1,1,0)	1	10	2.03	3596	3.596
U0/V0(1,1,1)	1	10	14.83	33962	33.962
V0(1,1,0)	1	10	14.92	33960	33.960
U0(1,1,0)	2	10	1.95	3596	3.596
U0/V0(1,1,1)	2	10	2.63	5184	5.184
V0(1,1,0)	2	10	2.60	4969	4.969
U0(1,1,0)	3	10	2.03	3596	3.596
U0/V0(1,1,1)	3	10	2.81	5196	5.196
V0(1,1,0)	3	10	2.68	4972	4.972
U0(1,1,0)	4	10	1.95	3596	3.596
U0/V0(1,1,1)	4	10	12.68	28644	28.644
V0(1,1,0)	4	10	12.72	28648	28.648

Discussion

The stabilizing feature of energy conserving methods can again be seen for this problem. The only one of the three algorithms which never conserves energy exactly, the Newmark Method (U0(1,1,0)), is seen to unstable in the Lagrange multipliers, acceleration, and energy regardless of which option is used. Around 14 seconds the system enters a state where a great deal of energy is stored in the flexible springs and oscillations grow unbounded. Similarly, the when the interpolation parameters are outside on both terms (option 1) we also see failure in accelerations, Lagrange multipliers, and energy conservation regardless of the base algorithm.

Interestingly, each of the remaining 6 algorithms: U0/V0(1,1,1) using option 2, 3, 4 and V0(1,1,0) using option 2, 3, and 4 are stable for very long term solutions. Each of these was was run out to 50 seconds and each maintained energy conservation as well as stability in accelerations and Lagrange multipliers for both $\Delta t = 0.01$ and $\Delta t = 0.05$. We see from the computational time table though that option 4 is highly undesirable compared to options 2 and 3 due to significantly more iterations to convergence. Options 2 and 3 are very comparable

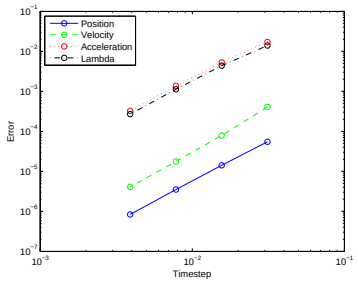
in both CPU time and iterations per step, with only a very slight edge to option 2.

Overall the results of this problem coincide closely with that from the rigid body problem. In both cases the V0(1,1,0) algorithm (midpoint rule with midpoint acceleration) using option 2 is optimal. The algorithm is able to conserve energy exactly as well as provide very long duration simulations without becoming unstable in any parameter.

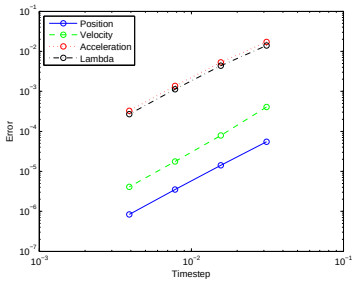
Convergence

As in the double pendulum example to ensure the order of accuracy of all algorithms remains, as designed, second order, convergence plots corresponding to each simulation above were created.

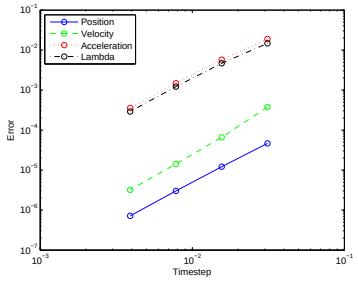
To create the convergence plots the problem was run to an end time of 0.25 seconds with the same parameters and tolerances as above. Again, the "exact" solution was obtained using $\Delta t_{exact} = 2^{-10}$. The four additional numerical solutions were run in order to calculate error using $\Delta t_1 = 2^{-8}$, $\Delta t_2 = 2^{-7}$, $\Delta t_3 = 2^{-6}$, and $\Delta t_4 = 2^{-5}$. Figs. 4.37 and 4.38 demonstrate that these methods maintain the intended accuracy even with the addition of the internal force term.



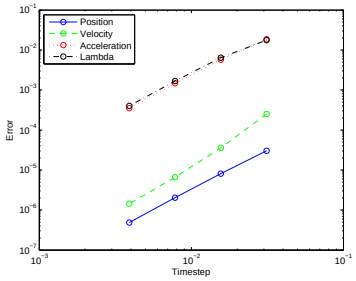
(a) Option 1 U0(1,1,0)



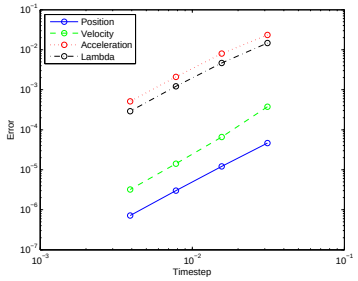
(b) Option 2 U0(1,1,0)



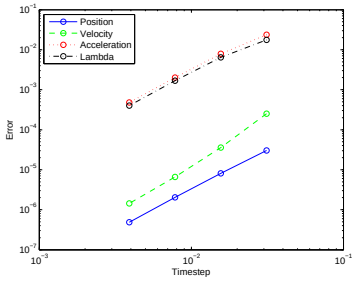
(c) Option 1 U0/V0(1,1,1)



(d) Option 2 U0/V0(1,1,1)

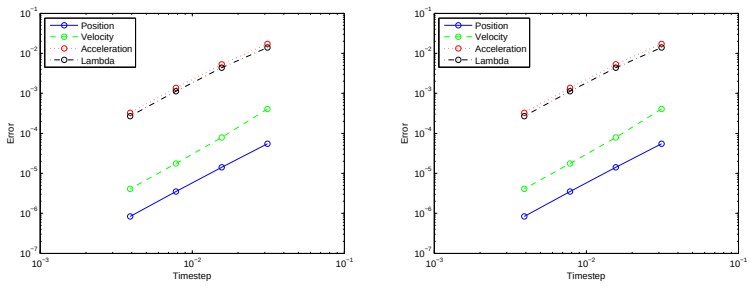


(e) Option 1 V0(1,1,0)



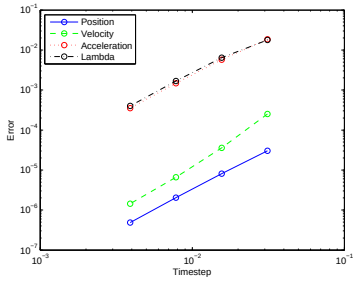
(f) Option 2 V0(1,1,0)

Figure 4.37: Four Mass System - Option 1, 2 Convergence

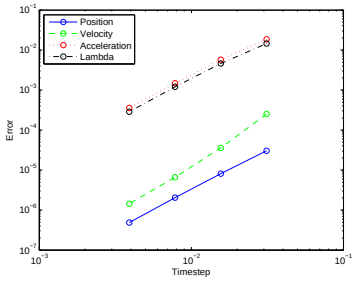


(a) Option 3 U0(1,1,0)

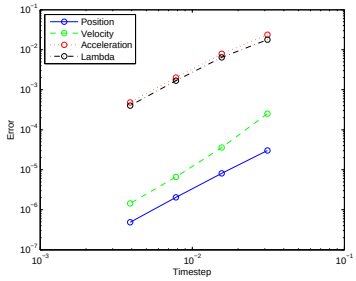
(b) Option 4 U0(1,1,0)



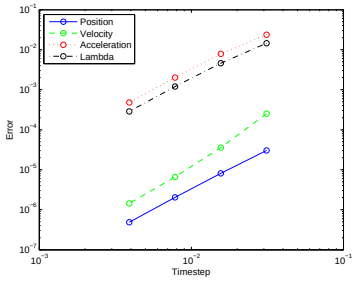
(c) Option 3 U0/V0(1,1,1)



(d) Option 4 U0/V0(1,1,1)



(e) Option 3 V0(1,1,0)

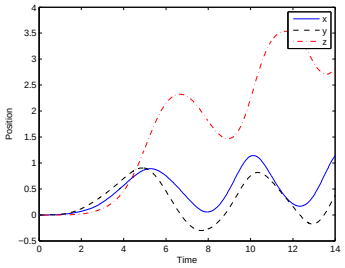


(f) Option 4 V0(1,1,0)

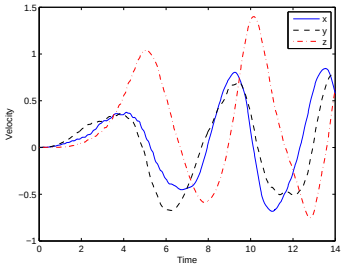
Figure 4.38: Four Mass System - Option 3, 4 Convergence

Two Field Form Results

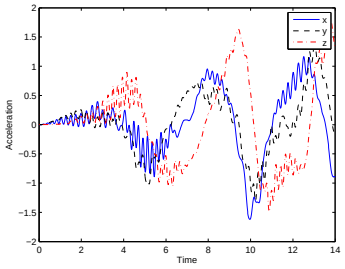
The two field algorithm in its v-form representation (equivalent to V0(1,1,0) algorithm) results are shown in Figs. 4.39-4.42. All simulation parameters are identical to the single field form simulations above.



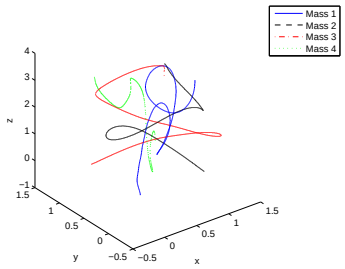
(a) Mass 1 Position



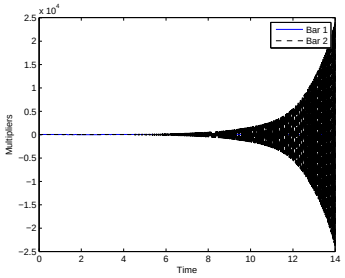
(b) Mass 1 Velocity



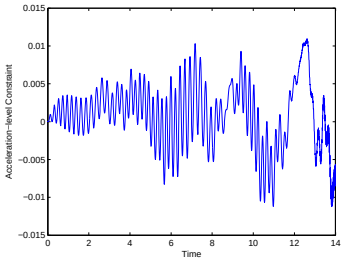
(c) Mass 1 Acceleration



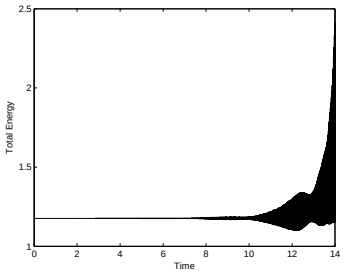
(d) Trajectories



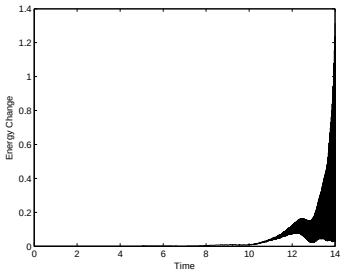
(e) Multipliers



(f) Acceleration Constraint

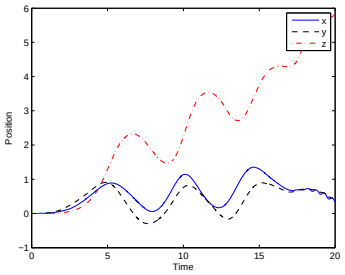


(g) Total Energy

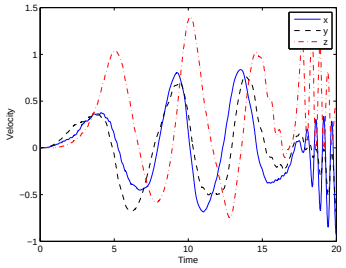


(h) Energy Change

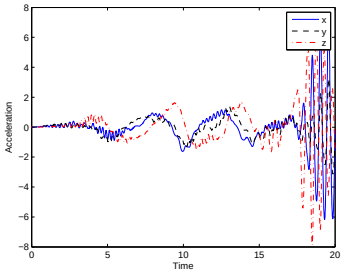
Figure 4.39: Four Mass System - Option 1 - Two Field Form



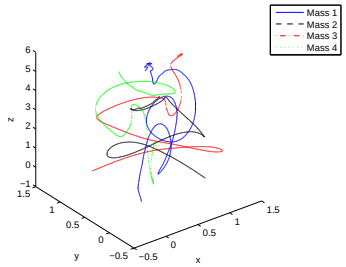
(a) Mass 1 Position



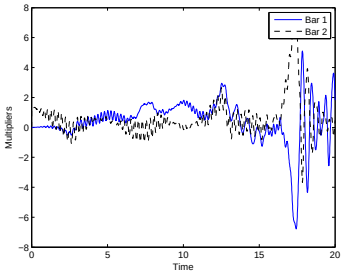
(b) Mass 1 Velocity



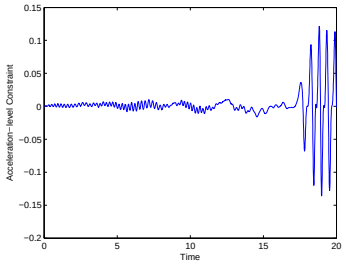
(c) Mass 1 Acceleration



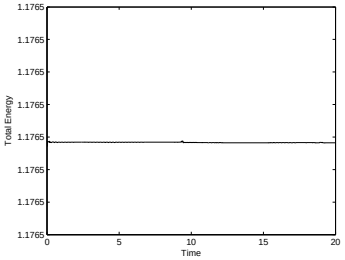
(d) Trajectories



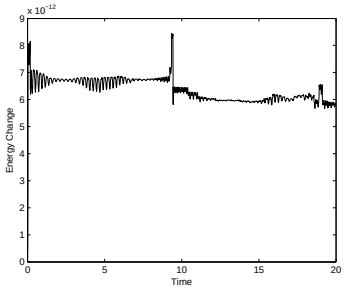
(e) Multipliers



(f) Acceleration Constraint

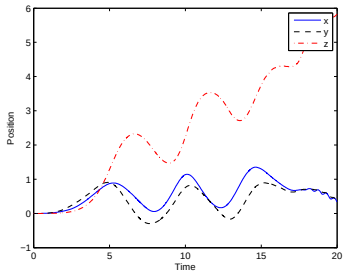


(g) Total Energy

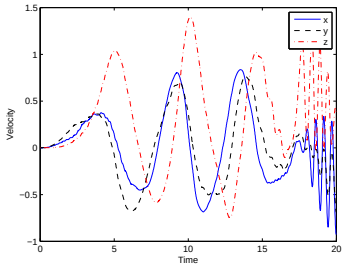


(h) Energy Change

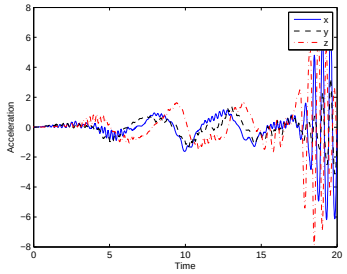
Figure 4.40: Four Mass System - Option 2 - Two Field Form



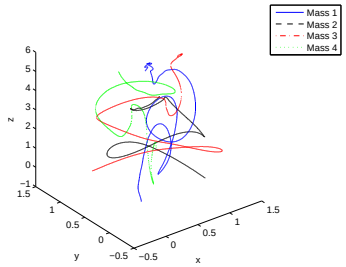
(a) Mass 1 Position



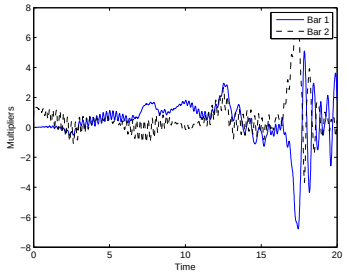
(b) Mass 1 Velocity



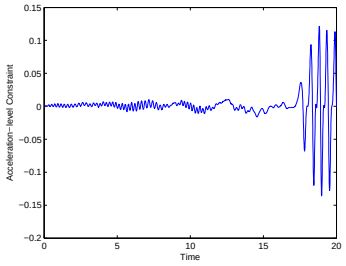
(c) Mass 1 Acceleration



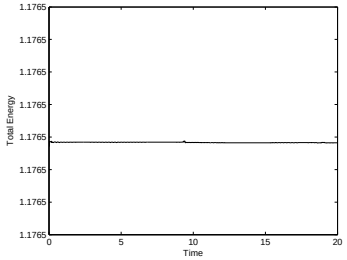
(d) Trajectories



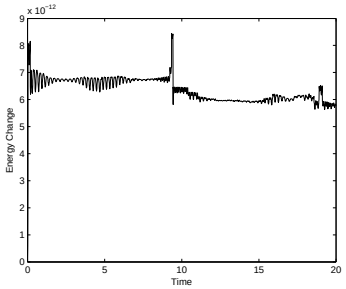
(e) Multipliers



(f) Acceleration Constraint

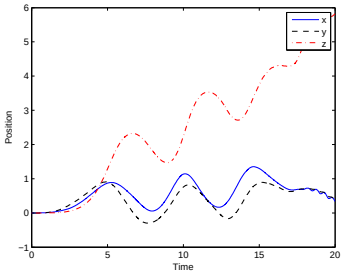


(g) Total Energy

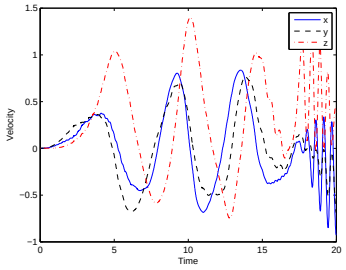


(h) Energy Change

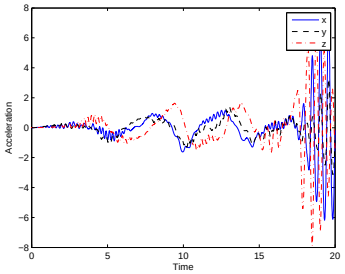
Figure 4.41: Four Mass System - Option 3 - Two Field Form



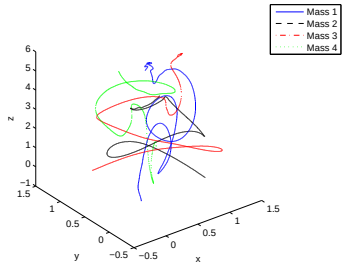
(a) Mass 1 Position



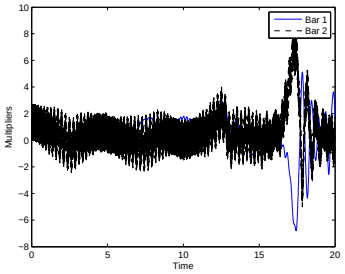
(b) Mass 1 Velocity



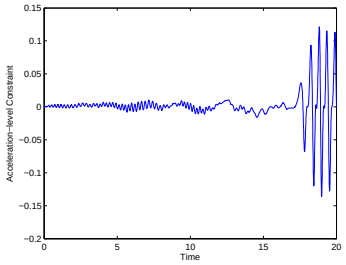
(c) Mass 1 Acceleration



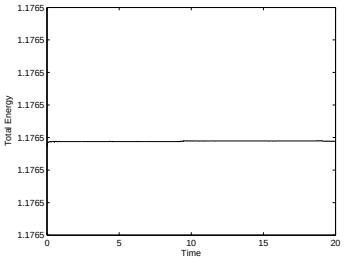
(d) Trajectories



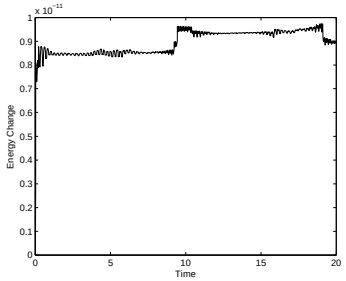
(e) Multipliers



(f) Acceleration Constraint



(g) Total Energy



(h) Energy Change

Figure 4.42: Four Mass System - Option 4 - Two Field Form

4.2.2 Thin Beam System

The final test problem consists of a simplified model for thin beams which combines bending and axial deformations. Armero [4] introduced this problem as demonstration of the need for controllable numerical dissipation in problems which combine rigid and flexible components within the same model. This problem extends the methodology of the above examples to a further generalization of energy conserving algorithms. The thin beams are modeled based on a nonlinear potential which requires careful treatment of the internal force to maintain energy conservation. The energy conserving method outlined by Armero is based on the method introduced by Greenspan in [7].

To demonstrate the algorithms developed herein, the rigid links in the model are treated as analytic rigid by constraint equations in the index 3 DAE system. The system is fundamentally a planar four-bar mechanism in which the two links pinned to ground are treated as rigid. The coupler link connecting the two rigid links is a simplified thin beam model consisting of an axial potential between any two nodes and a bending potential between any three nodes. In this case the coupler link is made up of five nodes. As such, there are 4 axial elements and 3 bending elements which comprise the link. The system layout and element types can be seen in Figs. 4.43 and 4.44.

The overall system under consideration has 10 degrees of freedom and two constraint equations (thus, two Lagrange multipliers). The degrees of freedom and constraint equations are as follows:

$$\mathbf{u} = \begin{bmatrix} x_1 & y_1 & x_2 & y_2 & x_3 & y_3 & x_4 & y_4 & x_5 & y_5 \end{bmatrix}^T \tag{4.37}$$

$$\dot{\mathbf{u}} = \begin{bmatrix} \dot{x}_1 & \dot{y}_1 & \dot{x}_2 & \dot{y}_2 & \dot{x}_3 & \dot{y}_3 & \dot{x}_4 & \dot{y}_4 & \dot{x}_5 & \dot{y}_5 \end{bmatrix}^T \tag{4.38}$$

$$\ddot{\mathbf{u}} = \begin{bmatrix} \ddot{x}_1 & \ddot{y}_1 & \ddot{x}_2 & \ddot{y}_2 & \ddot{x}_3 & \ddot{y}_3 & \ddot{x}_4 & \ddot{y}_4 & \ddot{x}_5 & \ddot{y}_5 \end{bmatrix}^T \tag{4.39}$$

$$\boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 & \lambda_2 \end{bmatrix}^T \tag{4.40}$$

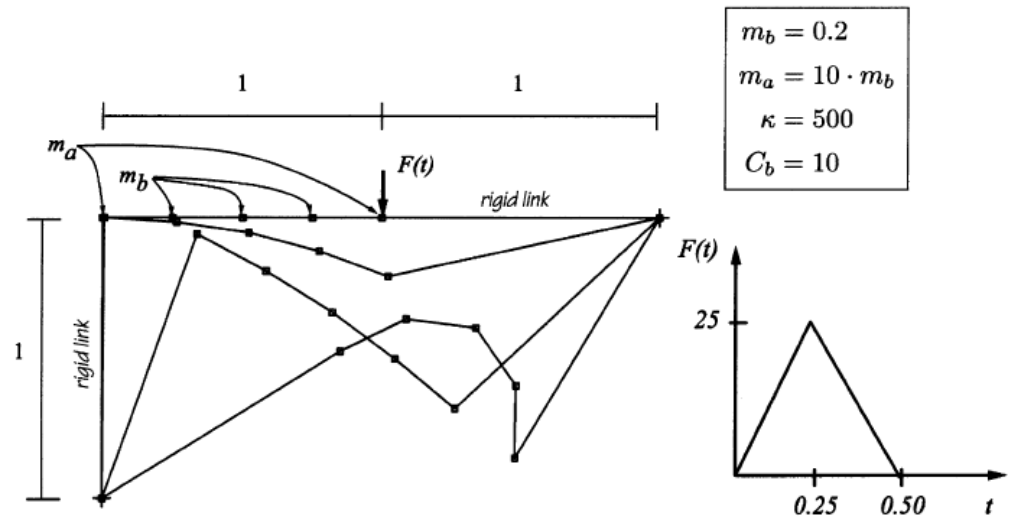


Figure 4.43: Thin Beam System from [4]

$$\mathbf{\Phi} = \begin{bmatrix} 1 - (x_1)^2 - (y_1)^2 \\ 1 - (2 - x_5)^2 - (1 - y_5)^2 \end{bmatrix} \tag{4.41}$$

The system is initially at rest, with a triangular force pulse applied over the first 0.5 seconds. The elements are defined in terms of potential energy functions as follows:

$$V_{bend}(\psi, \nu) = \frac{1}{2} C_b \frac{\nu + \psi}{\nu - \psi}, \psi = \mathbf{r}_1 \cdot \mathbf{r}_2, \nu = l_1 l_2 \tag{4.42}$$

$$V_{axial}(l) = \frac{1}{2} \kappa (l - l_0)^2 \tag{4.43}$$

To obtain the nonlinear internal forces for each element, $\mathbf{P}$, we use the discrete energy-momentum method of [7] for the spring and beam elements as follows:

$$\mathbf{P}_{axial} = \frac{V_{axial}(l_{n+1}) - V_{axial}(l_n)}{l_{n+1} - l_n} \frac{\mathbf{u}_{n+1} + \mathbf{u}_n}{l_{n+1} + l_n} \tag{4.44}$$

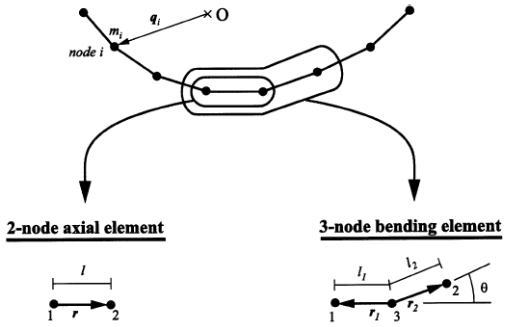


Figure 4.44: Element Description from [4]

$$\mathbf{P}_{bend} = \frac{1}{2}\frac{V_{bend}^{(n+1,n+1)} + V_{bend}^{(n+1,n)} - V_{bend}^{(n,n+1)} - V_{bend}^{(n,n)}}{\psi_{n+1} - \psi_n} \begin{bmatrix} \mathbf{r}_2 \\ \mathbf{r}_1 \\ -\mathbf{r}_1 - \mathbf{r}_2 \end{bmatrix}_{n+\frac{1}{2}}$$
$$+\frac{1}{2}\frac{V_{bend}^{(n+1,n+1)} + V_{bend}^{(n+1,n)} - V_{bend}^{(n,n+1)} - V_{bend}^{(n,n)}}{\nu_{n+1} - \nu_n} \left(\frac{l_2}{l_1} \begin{bmatrix} \mathbf{r}_1 \\ 0 \\ -\mathbf{r}_1 \end{bmatrix} + \frac{l_1}{l_2} \begin{bmatrix} 0 \\ \mathbf{r}_2 \\ -\mathbf{r}_2 \end{bmatrix} \right)_{n+\frac{1}{2}} \tag{4.45}$$

where the superscript of $V_{bend}(\psi, \nu)$ is the time level which ψ and ν are to be evaluated. That is, $V_{bend}^{(n+1,n)} = V_{bend}(\psi_{n+1}, \nu_n)$ and so on. It is worth noting that in the limiting case where the denominator is zero (at minimum the first iteration of each time step) for $\mathbf{P}_{bend}$ or $\mathbf{P}_{axial}$ the internal force is instead evaluated simply by $V'_{n+\frac{1}{2}}$.

The equation of motion is assembled then into the familiar form:

$$\mathbf{M}\widetilde{\ddot{\mathbf{u}}} + \mathbf{P}(\widetilde{\mathbf{u}}) + \widetilde{\mathbf{B}^T\boldsymbol{\lambda}} = \widetilde{\mathbf{F}} \tag{4.46}$$

with mass matrix and constraint jacobian defined as:

$$\mathbf{M} = diag(\begin{bmatrix} 2.0 & 2.0 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 2.0 & 2.0 \end{bmatrix}) \tag{4.47}$$

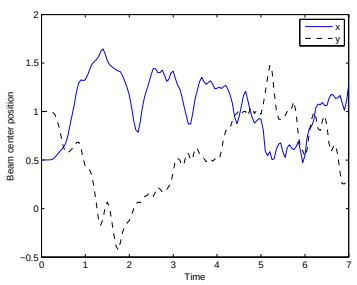
$$\mathbf{B}(\mathbf{u}) = \begin{bmatrix} -2x_1 & -2y_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2(2-x_5) & 2(1-y_5) \end{bmatrix} \tag{4.48}$$

Due to the specialized nature of the internal force required for energy conservation this system was tested using only the two algorithms which are consistent in the time level $n+\frac{1}{2}$: U0/V0(1,1,1) and V0(1,1,0). The U0(1,1,0) algorithm has an inherent time level of $n+1$ and has already been excluded as a stable method by the previous numerical examples.

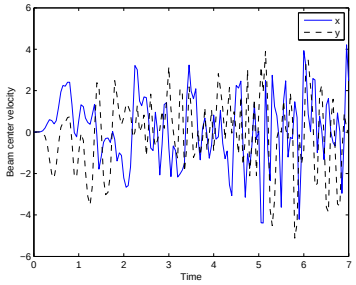
This system was run using a time step size of $\Delta t = 0.05$ and a nonlinear iteration tolerance of 10^{-6} using the d-form representation of the single field form algorithm. The system is initially at rest with the initial conditions shown below. The system response for the algorithms under consideration are shown in Figs. 4.45-4.52. As previously, a discussion of the results follows the figures.

$$\mathbf{u_0} = \begin{bmatrix} 0 & 1 & 0.25 & 1 & 0.5 & 1 & 0.75 & 1 & 1 & 1 \end{bmatrix}^T \tag{4.49}$$

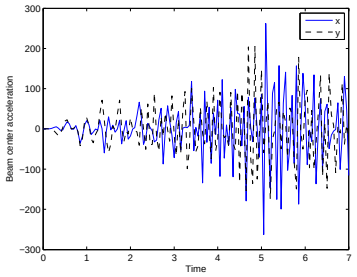
$$\dot{\mathbf{u}}_\mathbf{0} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{4.50}$$



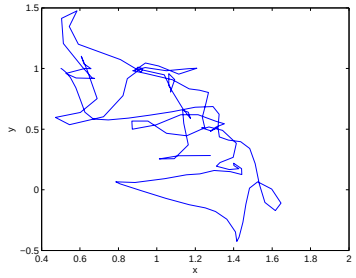
(a) Beam Center Position



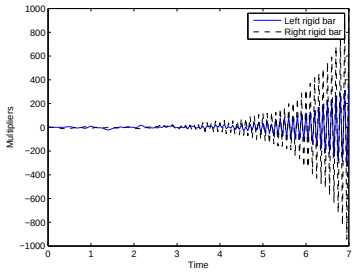
(b) Beam Center Velocity



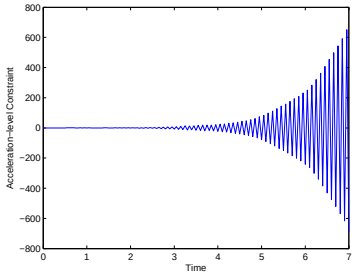
(c) Beam Center Acceleration



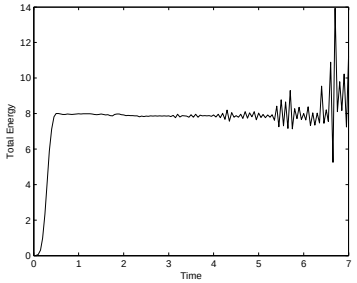
(d) Trajectories



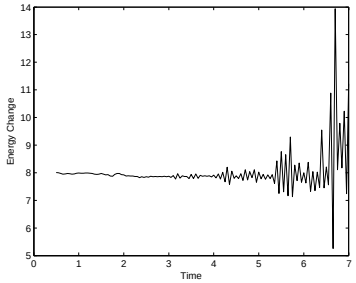
(e) Multipliers



(f) Acceleration Constraint

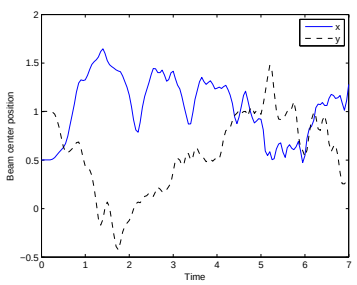


(g) Total Energy

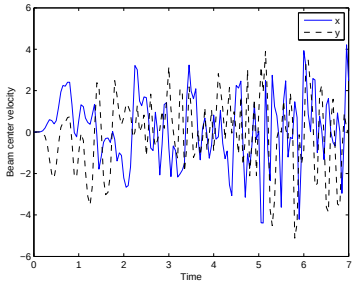


(h) Energy Change

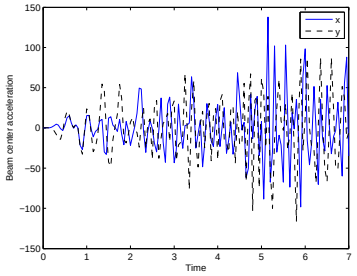
Figure 4.45: Thin Beam System - Option 1 - V0(1,1,1)



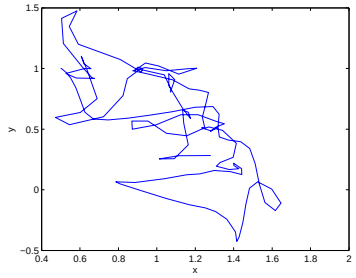
(a) Beam Center Position



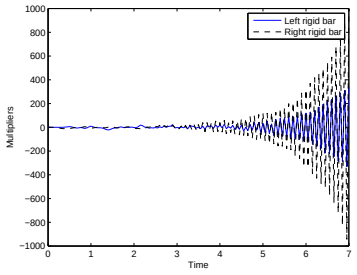
(b) Beam Center Velocity



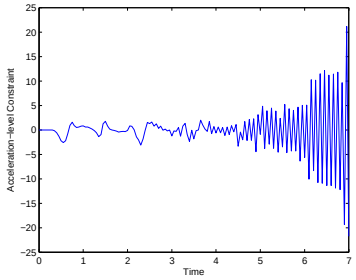
(c) Beam Center Acceleration



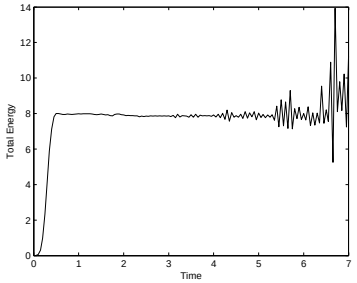
(d) Trajectories



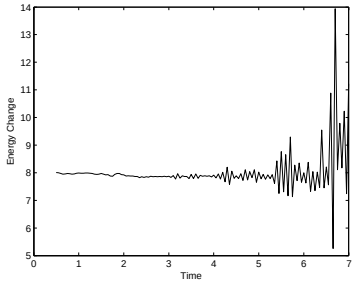
(e) Multipliers



(f) Acceleration Constraint

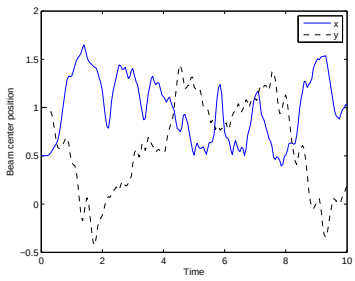


(g) Total Energy

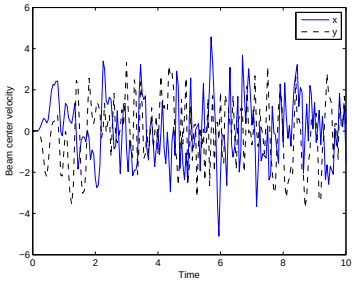


(h) Energy Change

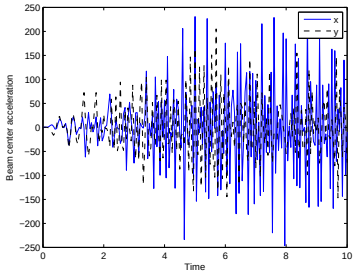
Figure 4.46: Thin Beam System - Option 1 - V0(1,1,0)



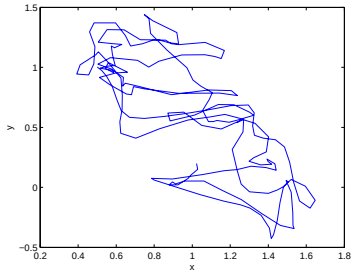
(a) Beam Center Position



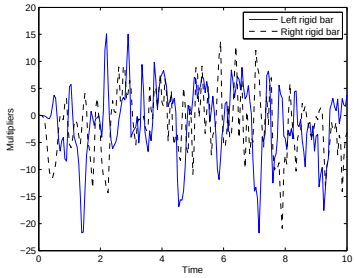
(b) Beam Center Velocity



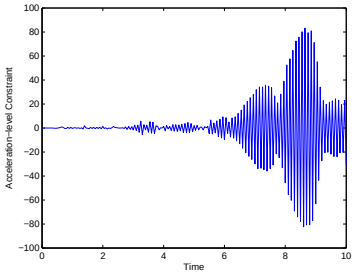
(c) Beam Center Acceleration



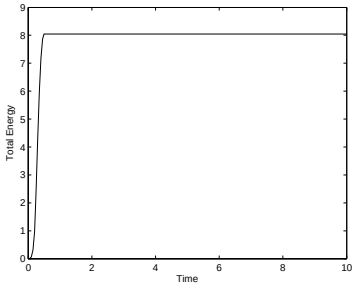
(d) Trajectories



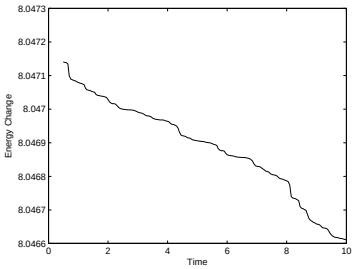
(e) Multipliers



(f) Acceleration Constraint

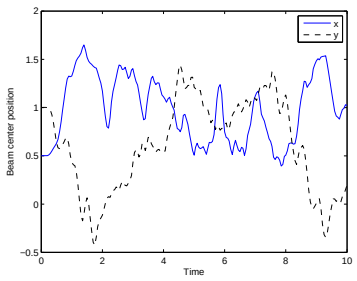


(g) Total Energy

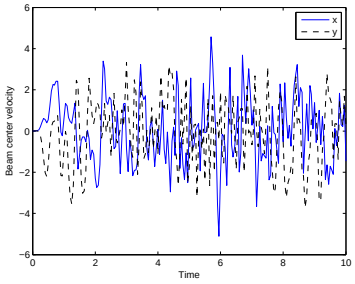


(h) Energy Change

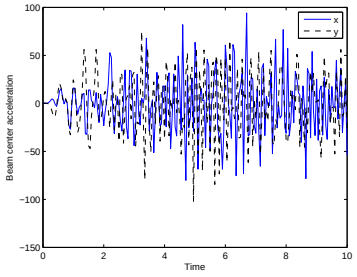
Figure 4.47: Thin Beam System - Option 2 - V0(1,1,1)



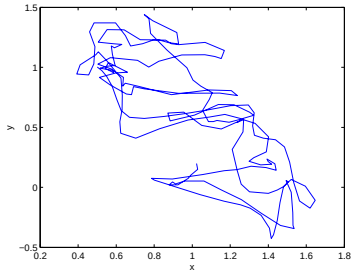
(a) Beam Center Position



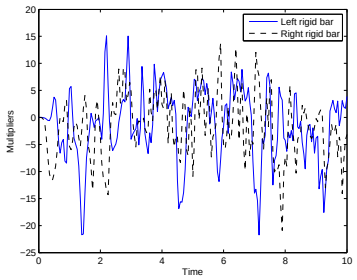
(b) Beam Center Velocity



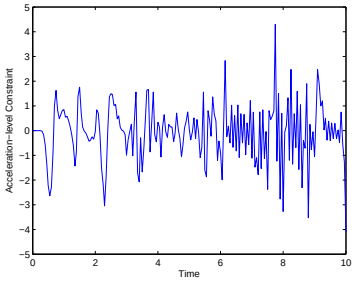
(c) Beam Center Acceleration



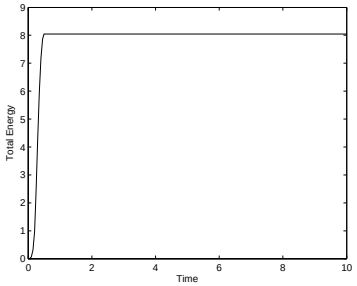
(d) Trajectories



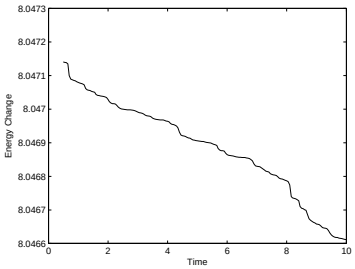
(e) Multipliers



(f) Acceleration Constraint

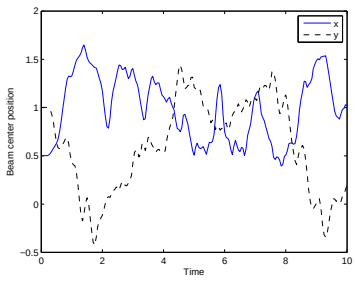


(g) Total Energy

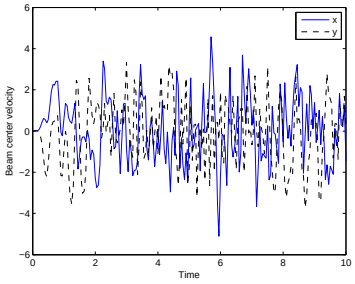


(h) Energy Change

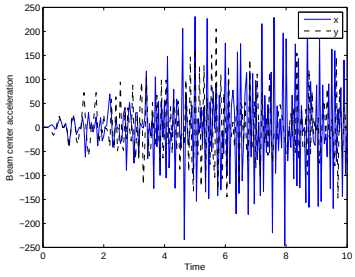
Figure 4.48: Thin Beam System - Option 2 - V0(1,1,0)



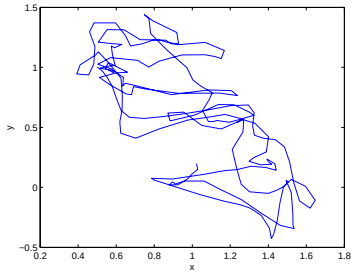
(a) Beam Center Position



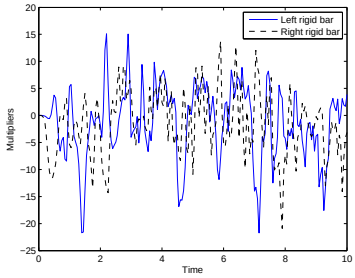
(b) Beam Center Velocity



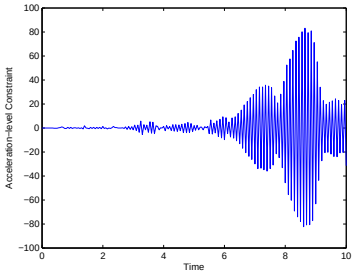
(c) Beam Center Acceleration



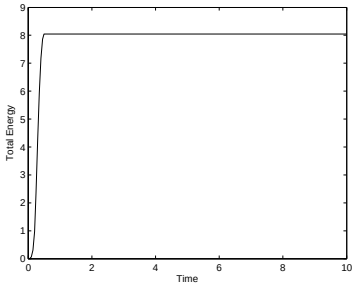
(d) Trajectories



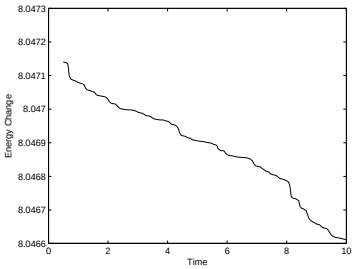
(e) Multipliers



(f) Acceleration Constraint

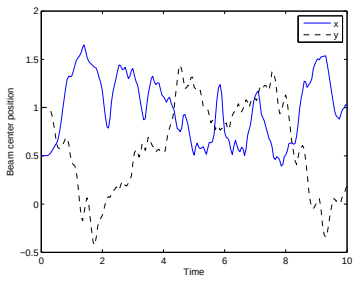


(g) Total Energy

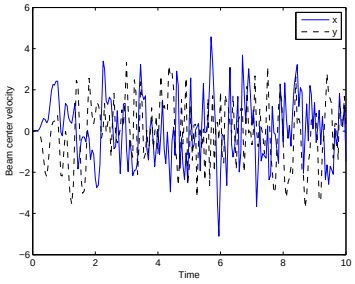


(h) Energy Change

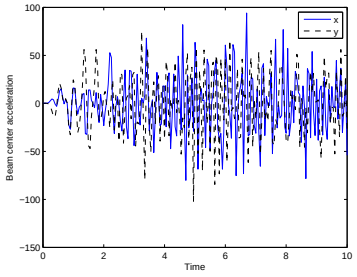
Figure 4.49: Thin Beam System - Option 3 - V0(1,1,1)



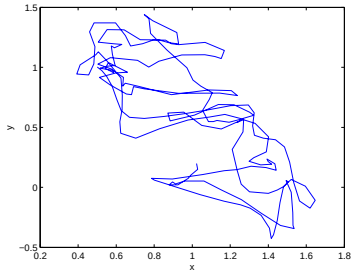
(a) Beam Center Position



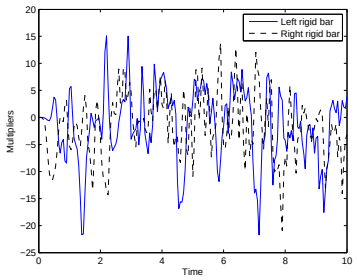
(b) Beam Center Velocity



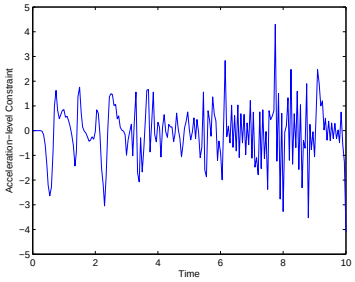
(c) Beam Center Acceleration



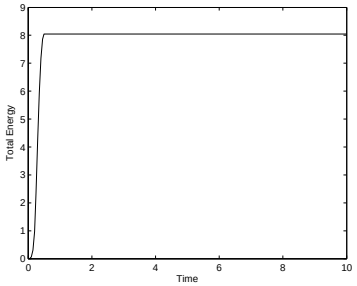
(d) Trajectories



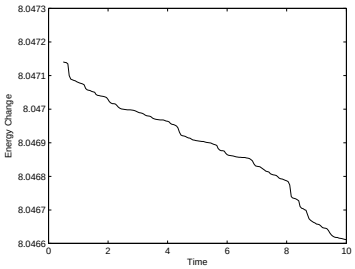
(e) Multipliers



(f) Acceleration Constraint

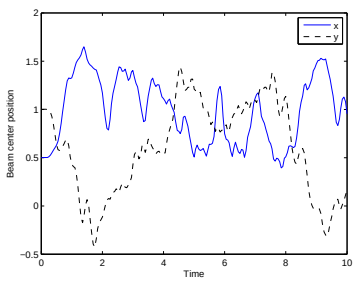


(g) Total Energy

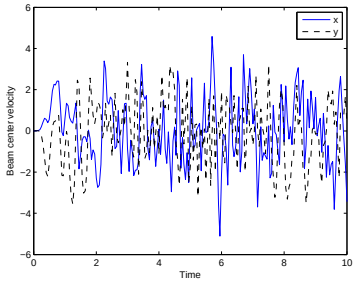


(h) Energy Change

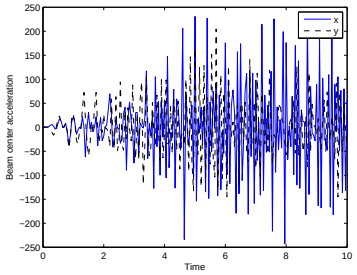
Figure 4.50: Thin Beam System - Option 3 - V0(1,1,0)



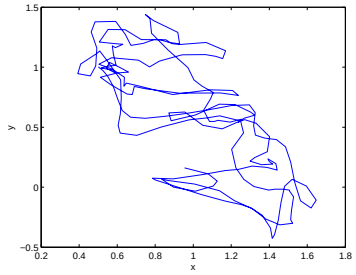
(a) Beam Center Position



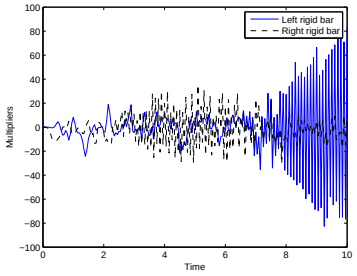
(b) Beam Center Velocity



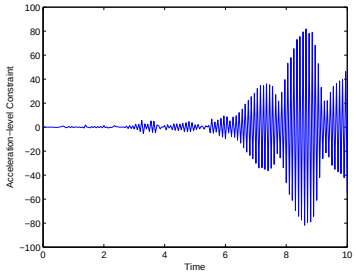
(c) Beam Center Acceleration



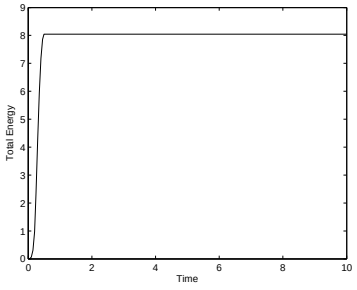
(d) Trajectories



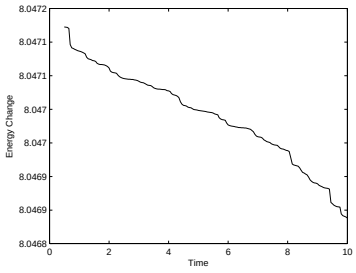
(e) Multipliers



(f) Acceleration Constraint

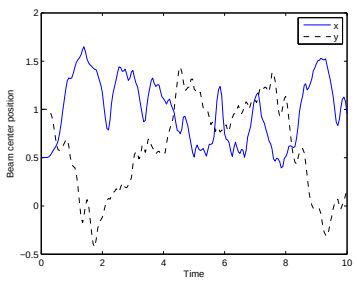


(g) Total Energy

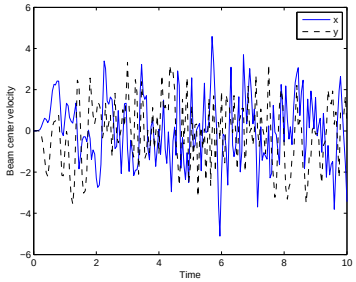


(h) Energy Change

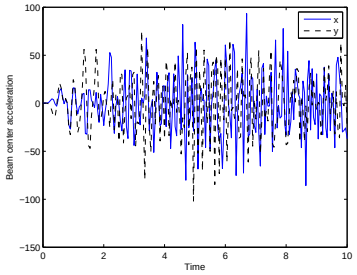
Figure 4.51: Thin Beam System - Option 4 - V0(1,1,1)



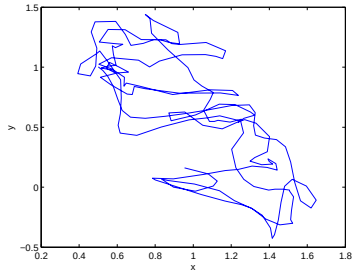
(a) Beam Center Position



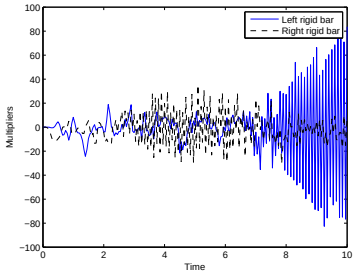
(b) Beam Center Velocity



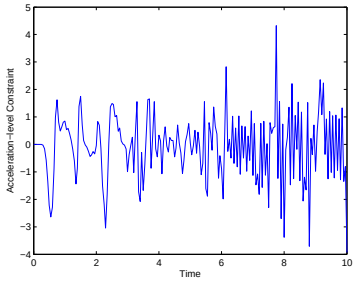
(c) Beam Center Acceleration



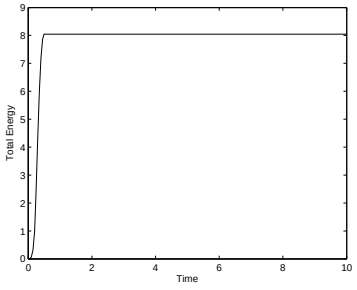
(d) Trajectories



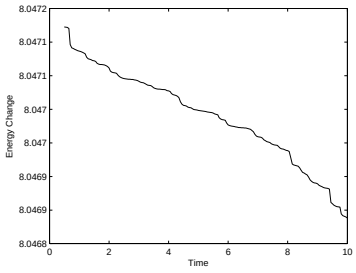
(e) Multipliers



(f) Acceleration Constraint



(g) Total Energy



(h) Energy Change

Figure 4.52: Thin Beam System - Option 4 - V0(1,1,0)

Discussion

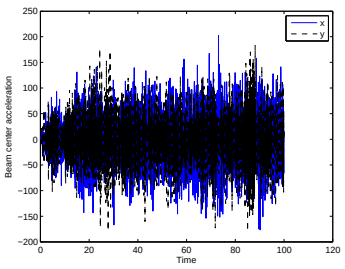
We again see here that using option 1, regardless of the base algorithm, the accelerations, Lagrange multipliers, and energy are unstable and the solution would soon fail due to nonlinear iteration convergence. Unlike the previous numerical examples, and likely due to the stiff nature of the problem, it can be seen that the Midpoint rule with endpoint accelerations (U0/V0(1,1,1)) has excessively large accelerations and very poor satisfaction of the acceleration level constraint equation regardless of the option 1, 2, 3, or 4. Again, these highly oscillatory accelerations lead to eventual convergence problems. Option 4 is also seen to be unsuitable for either base algorithm, but interestingly for the V0(1,1,0) algorithm shows drastic oscillation *only* in the Lagrange multipliers.

For the first time the nonlinear nature of the potential energy of the thin beam material make clear the claim of Betsch [17]: only evaluating $\mathbf{B}(\mathbf{u}_{n+\frac{1}{2}})$ is guaranteed to satisfy energy conservation. We see that using option 3 the energy is no longer conserved, meaning that the parameters outside method $\frac{1}{2}(\mathbf{B}(\mathbf{u}_n) + \mathbf{B}(\mathbf{u}_{n+1}))$ is insufficient for energy conservation in the nonlinear case. This rules out both base algorithms using option 3.

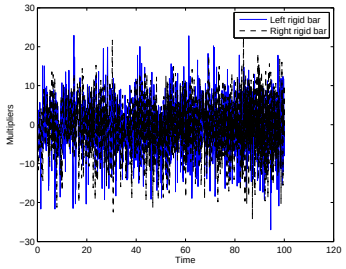
In this example, we see that the only algorithm which is perfectly stable and energy conserving is the V0(1,1,0) algorithm using option 2. This can also be referred to as the Midpoint rule with midpoint acceleration using parameters inside on both of the terms $\mathbf{B}(\mathbf{u})$ and $\boldsymbol{\lambda}$. This is quite an encouraging result, as the original paper of Armero uses this very stiff example as justification for the *need* of controllable numerical dissipation by showing that the energy conserving algorithm fails at around $t = 4$ seconds. To demonstrate the absolute stability of V0(1,1,0) - option 2 the problem was again run using the same parameters, but for a duration of 100 seconds. Fig. 4.53 demonstrates that even for extremely long duration simulation the acceleration, Lagrange multipliers, and energy are perfectly stable.

It has been shown through three numerical examples of varying type that the V0(1,1,0) algorithm using the option 2 methodology is the only scheme which

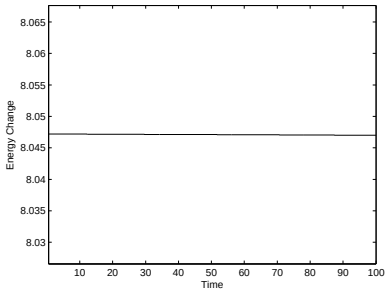
allows stable simulation of multibody dynamic systems of index 3 DAEs. This methodology is not only stable, but also second order accurate and energy conserving, eliminating any need for controllable numerical dissipation for these specific problems.



(a) Beam Center Acceleration



(b) Multipliers

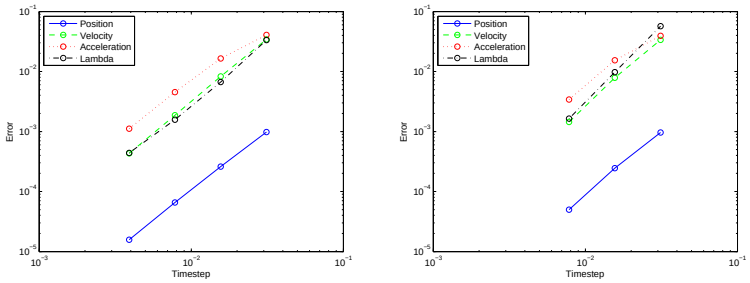


(c) Energy Change

Figure 4.53: Thin Beam Simulation to 100sec with V0(1,1,0) Option 2

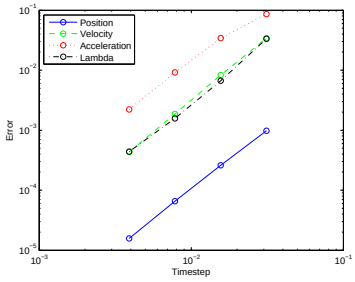
Convergence

Exactly as in the previous examples, convergence plots were generated for the problem with an end time of 0.25 seconds with the same parameters and tolerances as above. Again, the "exact" solution was obtained using $\Delta t_{exact} = 2^{-10}$. The test solutions were run using $\Delta t_1 = 2^{-8}$, $\Delta t_2 = 2^{-7}$, $\Delta t_3 = 2^{-6}$, and $\Delta t_4 = 2^{-5}$. Verification that all methods are of order Δt^2 is shown in Figs. 4.54 and 4.55.

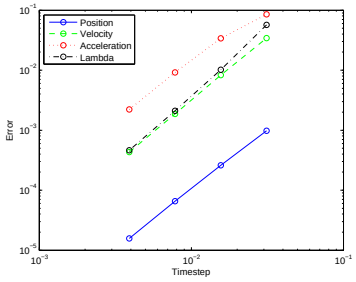


(a) Option 1 U0/V0(1,1,1)

(b) Option 2 U0/V0(1,1,1)

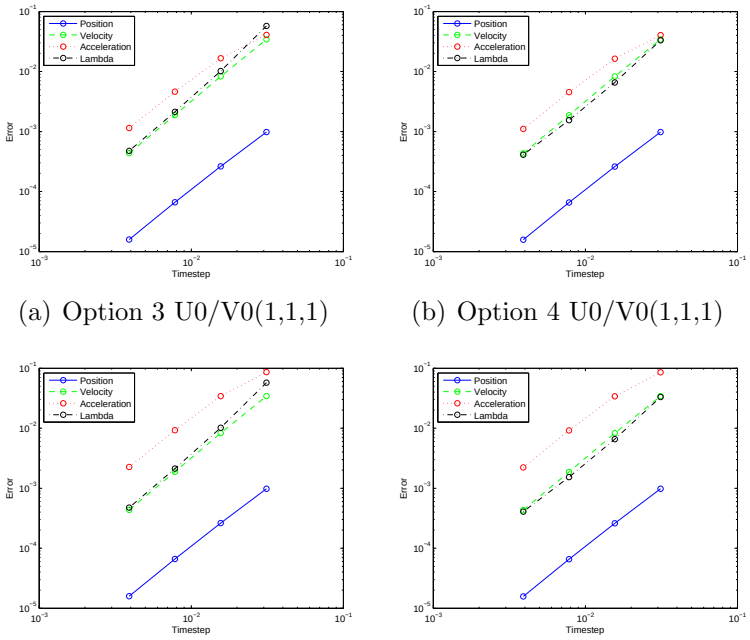


(c) Option 1 V0(1,1,0)



(d) Option 2 V0(1,1,0)

Figure 4.54: Thin Beam System - Option 1, 2 Convergence



(a) Option 3 U0/V0(1,1,1)

(b) Option 4 U0/V0(1,1,1)

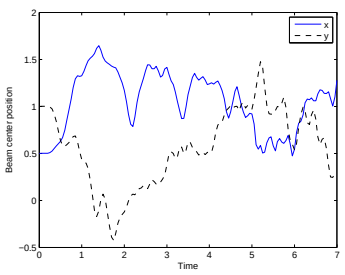
(c) Option 3 V0(1,1,0)

(d) Option 4 V0(1,1,0)

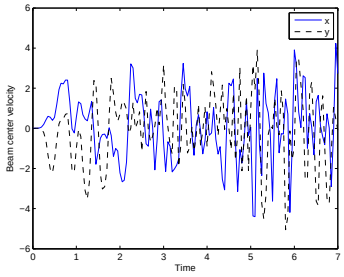
Figure 4.55: Thin Beam System - Option 3, 4 Convergence

Two Field Form Results

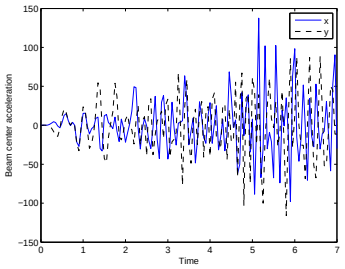
The two field algorithm in its v-form representation (equivalent to V0(1,1,0) algorithm) results are shown in Figs. 4.56-4.59. All simulation parameters are identical to the single field form simulations above.



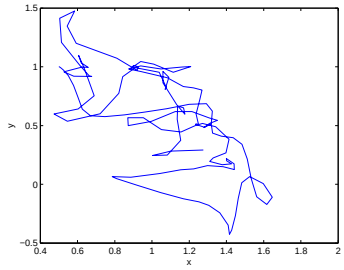
(a) Beam Center Position



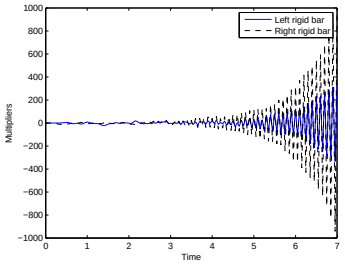
(b) Beam Center Velocity



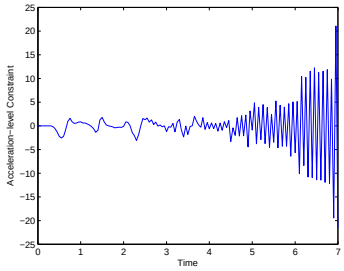
(c) Beam Center Acceleration



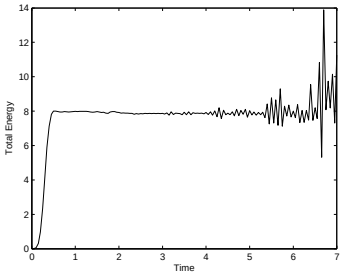
(d) Trajectories



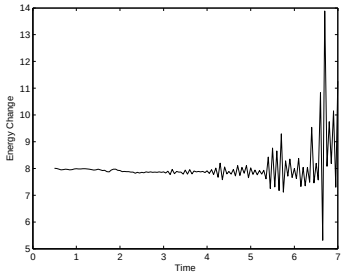
(e) Multipliers



(f) Acceleration Constraint

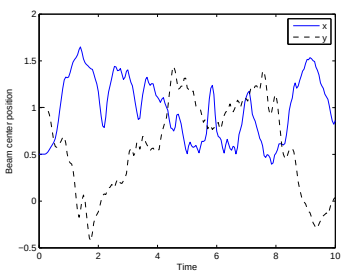


(g) Total Energy

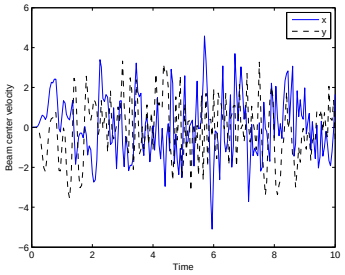


(h) Energy Change

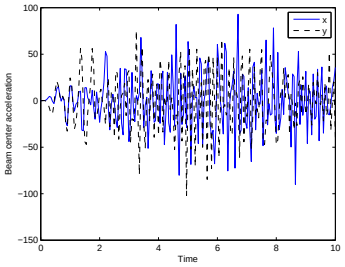
Figure 4.56: Thin Beam System - Option 1 - Two Field Form



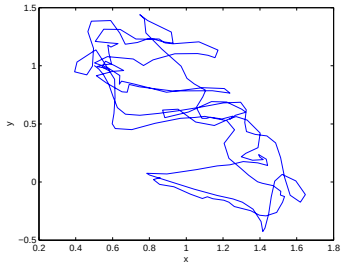
(a) Beam Center Position



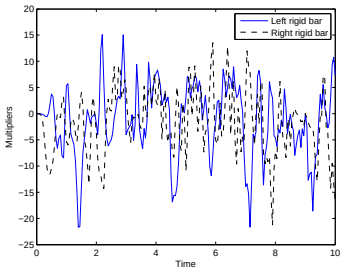
(b) Beam Center Velocity



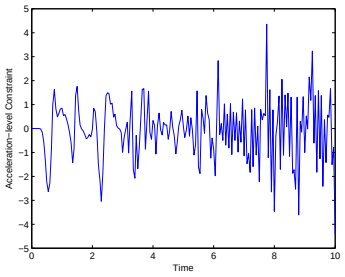
(c) Beam Center Acceleration



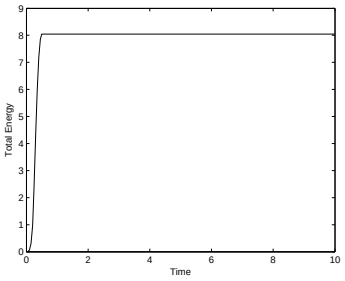
(d) Trajectories



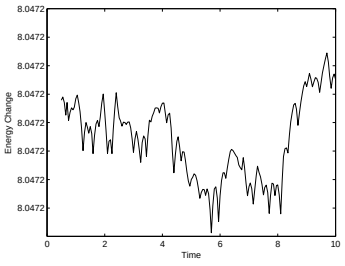
(e) Multipliers



(f) Acceleration Constraint

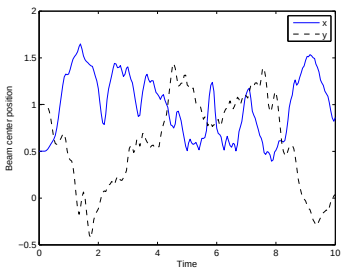


(g) Total Energy

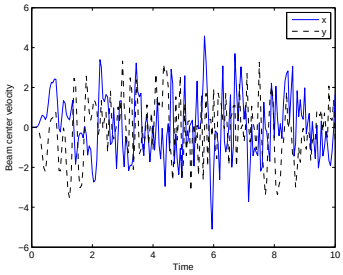


(h) Energy Change

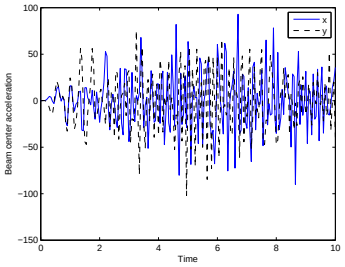
Figure 4.57: Thin Beam System - Option 2 - Two Field Form



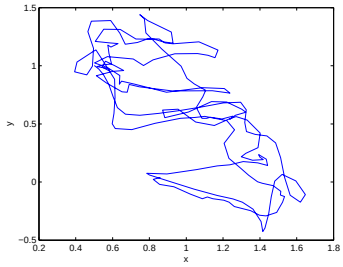
(a) Beam Center Position



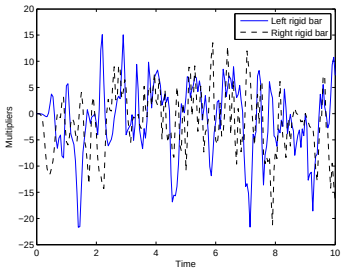
(b) Beam Center Velocity



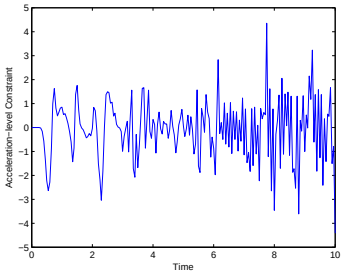
(c) Beam Center Acceleration



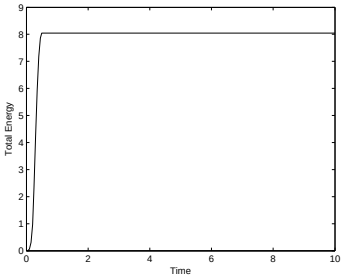
(d) Trajectories



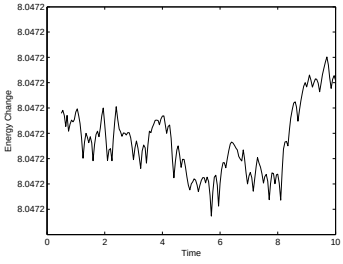
(e) Multipliers



(f) Acceleration Constraint

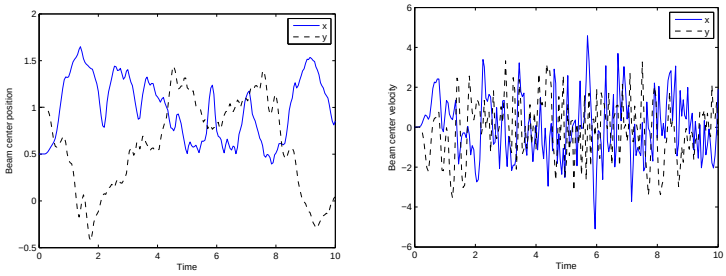


(g) Total Energy



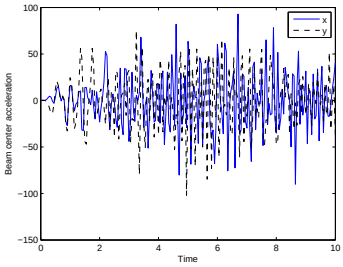
(h) Energy Change

Figure 4.58: Thin Beam System - Option 3 - Two Field Form

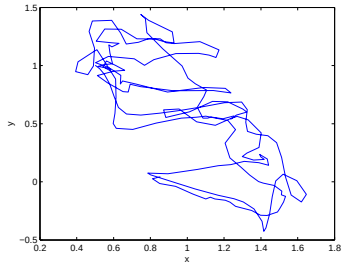


(a) Beam Center Position

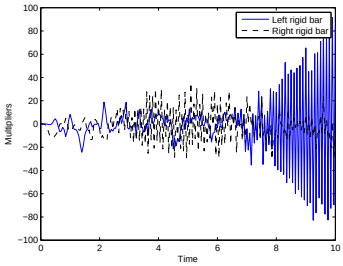
(b) Beam Center Velocity



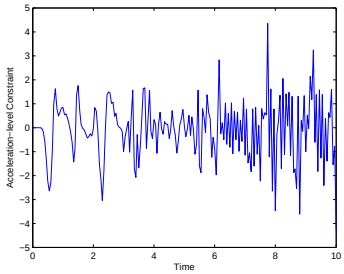
(c) Beam Center Acceleration



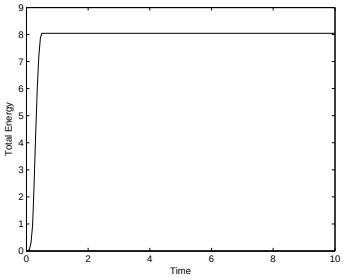
(d) Trajectories



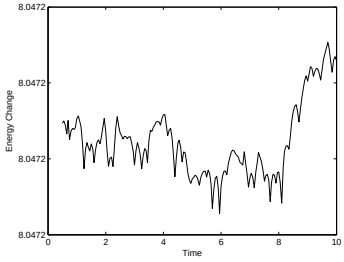
(e) Multipliers



(f) Acceleration Constraint



(g) Total Energy



(h) Energy Change

Figure 4.59: Thin Beam System - Option 4 - Two Field Form

Optimal Algorithm Constraint Satisfaction

To demonstrate the satisfaction of the position constraint and evaluation of the velocity and acceleration constraints the V0(1,1,0) algorithm was run for 2^{-3} seconds with a time step size of $\Delta t = 2^{-15}$. The results can be seen below.

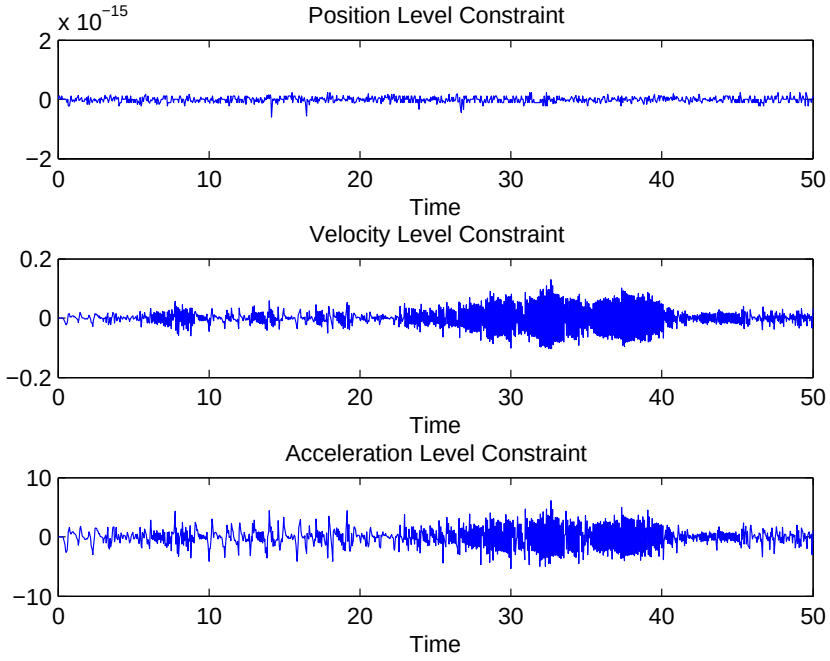


Figure 4.60: Thin Beam System - Optimal Algorithm Constraints

Chapter 5

Analysis and Conclusions

5.1 Analysis

In attempt to discover an algorithm capable of providing a stable, conserving, and accurate solution to the index 3 DAEs inherent to multibody dynamic systems the previous chapters employ the GSSSS family of algorithms as a general search space. Within this search space lies three non-dissipative algorithms: the Newmark method (U0(1,1,0)), the Midpoint rule with endpoint acceleration (U0/V0(1,1,1)), and the Midpoint rule with midpoint acceleration (V0(1,1,0)). Previous work on the application of these methods to nonlinear structural dynamics have provided two core concepts: the equation of motion time level and the normalized weighted residual method. Utilizing these concepts four distinct options are developed which extend the parent linear algorithms, designed to integrate ordinary differential equations, to the realm of nonlinear differential-algebraic equations.

As the intricacies of the normalized weighted residual procedure and the V0 family of algorithms are relatively unknown to the community at large, of the four options described herein only option 1 is prevalent in the literature. Of the twelve possible combinations, only the two U0 family algorithms can be readily found to be the focus of previous research: the Newmark method and the Midpoint

rule with endpoint acceleration. As, to date, a stable, conserving, and second order accurate algorithm cannot be found within the literature the remaining 10 algorithms (those which are fundamentally new to the community) were of primary interest.

To determine whether any of these algorithms were capable of overcoming the difficulties encountered when integrating the stiff differential-algebraic equations found in multibody systems three examples, all originally shown in the literature to fail, were tested. These examples are of increasing complexity. The first, the double pendulum, is a purely rigid body (or kinematic) system. The second, the four mass system, couples rigid bodies and geometrically nonlinear flexible bodies in the same model. Lastly, the thin beam system is comprised of a combination of rigid and flexible bodies with both geometric and material nonlinearities.

Examination of the results of the previous chapter yields the final conclusion that there *does* exist an algorithm able to provide stable, accurate, and conserving results, and that algorithm is option 2 V0(1,1,0). Of the twelve methods under examination the Midpoint rule with midpoint acceleration using the parameter inside approach for both $\mathbf{B}$ and $\boldsymbol{\lambda}$ was capable of overcoming all difficulties encountered by previous researchers.

The double pendulum, examined in [3], was originally shown to rapidly fail in nonlinear iteration convergence using the Newmark method. The author uses this failure to suggest the need for controllable numerical dissipation to foster long duration simulation of the problem. However, Fig. 4.15 demonstrates that using the V0(1,1,0) algorithm under option 2 not only is long duration simulation possible without the need for numerical dissipation, but it can conserve energy exactly.

The methodology presented in [42] for solution to the four mass system came very close to satisfying the desired requirements of an algorithm. The author, using a two field approach, shows a relatively long duration simulation as well as demonstrating second order accuracy in the displacements. It is however reported that the velocities and Lagrange multipliers suffer from a loss of accuracy.

Figs. 4.30 and 4.37 establish that the V0(1,1,0) option 2 algorithm is capable of long duration simulation, exact energy conservation, and second order accuracy in *all* primary variables (displacements, velocities, accelerations, and Lagrange multipliers).

The final example, that of the thin beam system, was originally proposed in [4] and shown to suffer from violent oscillations in accelerations which led to convergence failure. Again, the original author suggests the necessity of numerical dissipation to overcome these failures. Made clear in Figs. 4.48 and 4.53 is that the optimal algorithm discovered herein enables long duration simulation without acceleration oscillations as well as the ability to conserve energy exactly (a property lost upon using numerical dissipation).

For reference this optimal algorithm, V0(1,1,0) option 2, is written explicitly below.

$$\mathbf{M}\mathbf{a}_{n+1} + \nabla V(\mathbf{u}_{n+\frac{1}{2}}) + \mathbf{B}^T(\mathbf{u}_{n+\frac{1}{2}})\boldsymbol{\lambda}_{n+1} = F_{n+\frac{1}{2}} \tag{5.1}$$

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \dot{\mathbf{u}}_n\Delta t + \frac{1}{2}\ddot{\mathbf{u}}_n\Delta t^2 + \frac{1}{2}\Delta\mathbf{a}\Delta t^2 \tag{5.2}$$

$$\dot{\mathbf{u}}_{n+1} = \dot{\mathbf{u}}_n + \ddot{\mathbf{u}}_n\Delta t + \Delta\mathbf{a}\Delta t \tag{5.3}$$

$$\ddot{\mathbf{u}}_{n+1} = \ddot{\mathbf{u}}_n + \Delta\mathbf{a} \tag{5.4}$$

To obtain exactly energy conservation the ∇V term must be treated carefully for the material model under consideration. Shown in the previous chapters are two such methods. It is necessary to point out the seemingly inconsistent equation of motion time level in this algorithm. The subscript on the terms $\mathbf{a}_{n+1}$ and $\boldsymbol{\lambda}_{n+1}$ merely implies that these are the values which are returned from one step of the algorithm. Like the rest of the equation of motion, these terms also lie at $t_{n+\frac{1}{2}}$. The consequence of this shift on convergence plot creation is discussed in the previous chapter's convergence section for the double pendulum results.

An additional feature of this optimal algorithm is the lack of error in the initial time step acceleration calculations. Normally, any algorithm with a time level shift in acceleration suffers from difficulty in finding proper initial conditions for the

accelerations. Some authors have proposed a back-step to calculate the proper initial conditions while others ignore the (admittedly small) error. The V0(1,1,0) algorithm however does *not* suffer from this error in that the acceleration value of the previous time step does not enter the equations. That is, upon taking the first time step forward from the initial conditions, the initial value of $\mathbf{a}_0$ is completely unused by the algorithm except for an initial guess passed into the Newton iterations (and in this case the inconsistent acceleration at $t = 0$ is likely a better guess than time level shifted acceleration would be). As the Lagrange multipliers and accelerations are both shifted backward exactly one half time step, this benefit is also seen in the Lagrange multiplier term.

5.2 Conclusion

The primary goal of this research was to develop a robust algorithmic framework for integrating the index 3 differential algebraic equations natural to multibody dynamics. Currently the literature is filled with methods for solving these systems by index reduction and methods to overcome the addition problems introduced by index reduction. The present state-of-the-art requires reformulation of the naturally index 3 equations, demands additional computational effort, or necessitates usage of specialized stiff solvers. The methodology developed herein is able to overcome each of these previous obstacles.

This research was essentially a brute-force search of the GSSSS family of algorithms and the variety of methods for their extension to nonlinear differential-algebraic equations. Twelve possible methods were developed and demonstrated for three examples of differing physics. The significant result of this search is the V0(1,1,0) algorithm using the so-called option 2. Also termed the Midpoint rule with midpoint acceleration, this algorithm demonstrates all of the optimal features desired initially. These features include:

- Energy conservation

- Ability to foster long duration simulation without the need for controllable numerical dissipation
- Stable, non-oscillatory solutions for acceleration and Lagrange multipliers
- Direct application to the natural index 3 systems without the need for index reduction techniques
- Second order accuracy in displacement, velocity, acceleration, and Lagrange multipliers

To the knowledge of the author, no one method coupling all these features has been developed to-date. The great benefit of this method, and the reason previous research has sought stable index 3 methods, is that it allows existing codes to include constraints with almost trivial change to existing software. That is, existing codes for structural dynamics are able to include constrained dynamics without the need to entirely reformulate their software and without the need for a specialized solver. This method may provide a way to bridge the current gap between codes used to solve rigid multibody dynamics and those used to solve continuum systems, thus greatly increasing efficiency of multibody system design.

References

[1] S. S. Sandhu. Developments in robust numerical simulation techniques for flexible-rigid multibody dynamic systems. Masters thesis dissertation, Department of Mechanical Engineering, The University of Minnesota, 2003.

[2] M. Borri, L. Trainelli, and A. Croce. The embedded projection method: A general index reduction procedure for constrained system dynamics. *Computer Methods in Applied Mechanics and Engineering*, 195(50-51):6974 – 6992, 2006. Multibody Dynamics Analysis.

[3] M. Geradin and A. Cardona. *Flexible Multibody Dynamics: A Finite Element Approach*. John Wiley, Chichester ; New York, 2001.

[4] F. Armero and I. Romero. On the Formulation of High-frequency Dissipative Time-stepping Algorithms for Nonlinear Structural Dynamics. Part I: Low-order Methods for Two Model Problems and Nonlinear Elastodynamics. *Computer Methods in Applied Mechanics and Engineering*, 190:2603–2649, 2001.

[5] R. McLachlan. Six Lectures on Geometric Integration. *Foundations of Computational Mathematics*, pages 155–210, 2001.

[6] E. Noether. Invariante Variationsprobleme. In *Nachr. d. König. Gesellsch. d. Wiss. zu Göttingen*, pages 235–257, Math-Phys. Klasse, 1918.

[7] R. A. Labudde and D. Greenspan. Energy and Momentum Conserving Methods of Arbitary Order for the Numerical Integration of Equations of Motion Part I. *Numerisch Mathematik*, 25:323–346, 1976.

[8] R. A. Labudde and D. Greenspan. Energy and Momentum Conserving Methods of Arbitary Order for the Numerical Integration of Equations of Motion Part II. *Numerisch Mathematik*, 26:1–16, 1976.

[9] D. Greenspan. Conservative Numerical Methods for "$\dot{x} = fx$". *Journal of Computational Physics*, 56:28–41, 1984.

[10] T. J. R. Hughes, T. K. Caughey, and W. K. Liu. Finite Element Methods for Nonlinear Elastodynamics which Conserve Energy. *Journal of Applied Mechanics*, 45:366–370, 1978.

[11] D. Kuhl and E. Ramm. Constraint Energy Momentum Algorithm and its Application to Non-linear Dynamics of Shells. *Computer Methods in Applied Mechanics and Engineering*, 136:293–315, 1999.

[12] J. C. Simo, N. Tarnow, and K. K. Wong. Exact Energy-Momentum Conserving Algorithms and Symplectic Schemes for Nonlinear Dynamics. *Computer Methods in Applied Mechanics and Engineering*, 100:63–116, 1992.

[13] J. C. Simo and N. Tarnow. The Discrete Energy-Momentum Method. Part I. Conserving Algorithms for Nonlinear Elastodynamics. *ZAMP*, 43:757–793, 1992.

[14] K. Kuhl and M. Crisfield. Energy-Conserving and Decaying Algorithms in Non-linear Structural Dynamics. *International Journal for Numerical Methods in Engineering*, 45:569–599, 1999.

[15] S. U. Masuri. A Normalized Time Weighted Residual Approach for Nonlinear Structural Dynamics. Masters thesis dissertation, Department of Mechanical Engineering, The University of Minnesota, 2007.

[16] O. Gonzalez. Exact Energy and Momentum Conserving Algorithms for General Models in Nonlinear Elasticity. *Computer Methods in Applied Mechanics and Engineering*, 190:1763–1783, 2000.

[17] P. Betsch. The discrete null space method for the energy consistent integration of constrained mechanical systems: Part i: Holonomic constraints. *Computer Methods in Applied Mechanics and Engineering*, 194(50-52):5159 – 5190, 2005.

[18] K. K. Tamma, X. Zhou, and D. Sha. The Time Dimension: A Theory of Development/Evolution, Classification, Characterization and Design of Computational Algorithms for Transient/Dynamic Applications. *Archives in Computational Mechanics*, 7(2):67–290, 2000.

[19] K. K. Tamma, X. Zhou, and D. Sha. A Theory of Development and Design of Generalized Integration Operators for Computational Structural Dynamics. *International Journal of Numerical Methods in Engineering*, 50:1619–1664, 2001.

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

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

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

[23] K. K. Tamma and R. R. Namburu. Applicability and Evaluation of An Implicit Self-Starting Unconditionally Stable Methodology for Dynamics of Structures. *Computers and Structures*, 34:835–842, 1990.

[24] N. M. Newmark. A Method of Computation for Structural Dynamics. *Journal for American Society of Civil Engineers*, 1:67–94, 1959.

[25] W. L. Wood, M. Bossak, and O. C. Zienkiewicz. An Alpha Modification of Newmark's Method. *International Journal for Numerical Methods in Engineering*, 15:1562–1566, 1980.

[26] H. M. Hilber, T. J. R. Hughes, and R. L. Taylor. Improved Numerical Dissipation for Time Integration Algorithms in Structural Dynamics. *Earthquake Engineering and Structural Dynamics*, 5:283–292, 1977.

[27] A. Hoitink. Investigations Encompassing the Equations of Motion and Proper and Accurate Treatment of Algorithmic Variables: Computational Structural Dynamics and Stiff Systems. Masters thesis dissertation, Department of Mechanical Engineering, The University of Minnesota, 2009.

[28] A. Hoitink, S. Masuri, X. Zhou, and K. K. Tamma. Algorithms by design: Part i: On the hidden point collocation within lms methods and implications for nonlinear dynamics applications. *International Journal for Computational Methods in Engineering Science and Mechanics*, 9:383–407, 2008.

[29] S. U. Masuri, A. Hoitink, X. Zhou, and K. K. Tamma. Algorithms by design: Part iii: A novel normalized time weighted residual methodology and design of optimal symplectic-momentum based controllable numerical dissipative algorithms for nonlinear structural dynamics. *International Journal for Computational Methods in Engineering Science and Mechanics*, 10:57–90, January 2009.

[30] S. U. Masuri, A. Hoitink, X. Zhou, and K. K. Tamma. Algorithms by design: A new normalized time-weighted residual methodology and design of a family

of energymomentum conserving algorithms for nonlinear structural dynamics. *International Journal for Numerical Methods in Engineering*, 79(9):1094–1146, 2009.

[31] K.E. Brenan, S.L Campbell, and L. Petzold. *Numerical Solutions of Initial-Value Problems in Differential-Algebraic Equations*. North Holland, New York, 1989.

[32] E. Hairer, S. P. Norsett, and G. Wanner. *Solving Ordinary differential equations*, volume 8, 14. Springer-Verlag, Berlin ; New York, 1993; 1996.

[33] J. Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. *Computer Methods in Applied Mechanics and Engineering*, 1(1):1 – 16, 1972.

[34] C. Lubich. On projected runge-kutta methods for differential algebraic equations. *BIT Numerical Mathematics*, 31:545–550, 1991.

[35] L. R. Petzold. A description of dassl: A differential/algebraic system solver. In *International Conference on Scientific Computing*, 1983.

[36] C. W. Gear, B. Leimkuhler, and G. K. Gupta. Automatic integration of euler-lagrange equations with constraints. *Journal of Computational and Applied Mathematics*, 12-13:77 – 90, 1985.

[37] C. Lunk and B. Simeon. Solving constrained mechanical systems by the family of newmark and α-methods. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift fr Angewandte Mathematik und Mechanik*, 86(10):772–784, 2006.

[38] Laurent Jay and Dan Negrut. Extensions of the hht-α method to differential-algebraic equations in mechanics. *Electronic Transactions on Numerical Analysis*, 26:190–208, 2007.

[39] J. Yen, L. Petzold, and S. Raha. A time integration algorithm for flexible mechanism dynamics: The dae α-method. *Computer Methods in Applied Mechanics and Engineering*, 158(3-4):341 – 355, 1998.

[40] N. Khude, L. O. Jay, A. Schaffer, and D. Negrut. A discussion of low order numerical integration formulas for rigid and flexible multibody dynamics. *ASME Conference Proceedings*, 2007(4806X):149–160, 2007.

[41] C. Hoff, T.J.R. Hughes, G. Hulbert, and P.J. Pahl. Extended comparison of the hilber-hughes-taylor α-method and the θ1-method. *Computer Methods in Applied Mechanics and Engineering*, 76(1):87 – 93, 1989.

[42] O. Gonzalez. Mechanical systems subject to holonomic constraints: Differential algebraic formulations and conservative integration. *Physica D: Nonlinear Phenomena*, 132(1-2):165 – 174, 1999.

[43] C. Arevalo and P. Lotstedt. Improving the accuracy of bdf methods for index 3 differential-algebraic equations. *BIT Numerical Mathematics*, 35:297–308, 1995.

Appendix A

Additional Numerical Examples

A.1 Circle Track System

To validate a problem for which an exact solution is known, the circular track problem described in [43] is illustrated below. Physically synonymous to a rigid pendulum, a mass fixed to a circular track is given a constant force tangent to its path. The x- and y-coordinate of the mass describe the two degrees of freedom of the system with the center of the track at the origin. The equation of motion and constraint equation are defined as:

$$\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} + \begin{bmatrix} -2x \\ -2y \end{bmatrix} \boldsymbol{\lambda} = \begin{bmatrix} 4y \\ -4x \end{bmatrix} \tag{A.1}$$

$$\Phi = 1 - x^2 - y^2 \tag{A.2}$$

The exact solution of the problem can be derived as:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \sin(t^2) \\ \cos(t^2) \end{bmatrix} \tag{A.3}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 2t\cos(t^2) \\ -2t\sin(t^2) \end{bmatrix} \tag{A.4}$$

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} -4t^2 \sin(t^2) + 2\cos(t^2) \\ -4t^2 \cos(t^2) - 2\sin(t^2) \end{bmatrix} \tag{A.5}$$

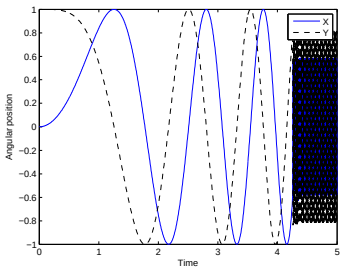
$$\boldsymbol{\lambda} = -4t^2 \tag{A.6}$$

The (2x2) $\mathbf{J}_{B\lambda}(\mathbf{u}, \boldsymbol{\lambda})$ matrix defined in Eqs. 4.15-4.18 for single field form and Eqn. 4.20-4.23 for two field form follows as:

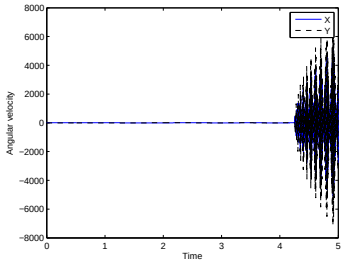
$$\begin{aligned} \mathbf{J}_{B\lambda}(1,1) &= -2\boldsymbol{\lambda} \\ \mathbf{J}_{B\lambda}(2,2) &= -2\boldsymbol{\lambda} \end{aligned} \tag{A.7}$$

Figs. A.1-A.12 verify that for this problem the same pattern emerges: the V0(1,1,0) algorithm using options 2, 3, and 4 provide stable solutions for the given time duration, with option 2 being optimal in terms for computational cost and long duration stability. Each run used a time step size of $\Delta t = 0.005$ and a nonlinear iteration tolerance of 10^{-8} using the a-form representation of the single field form algorithm.

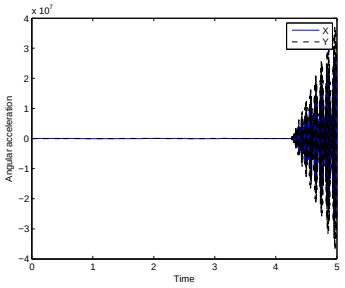
Convergence plots were generated with an end time of 2^{-2} seconds with the same parameters and tolerances as above. The numerical solutions were run using $\Delta t_1 = 2^{-4}$, $\Delta t_2 = 2^{-5}$, $\Delta t_3 = 2^{-6}$, and $\Delta t_4 = 2^{-7}$. Figs. A.13 and A.14 confirm precise second order accuracy regardless of the algorithm or option under consideration.



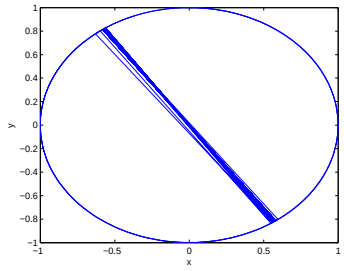
(a) Position



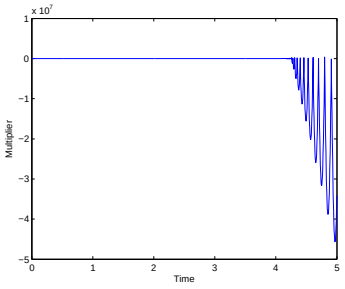
(b) Velocity



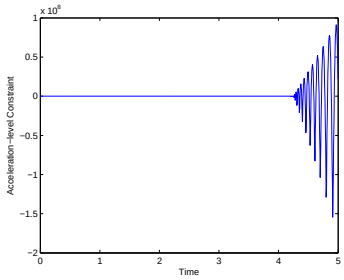
(c) Acceleration



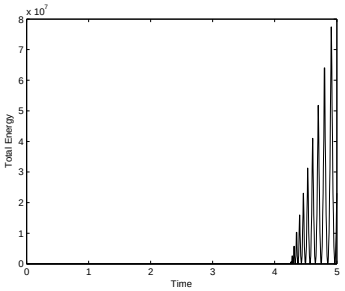
(d) Trajectory



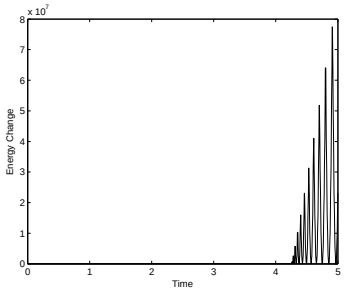
(e) Multiplier



(f) Acceleration Constraint

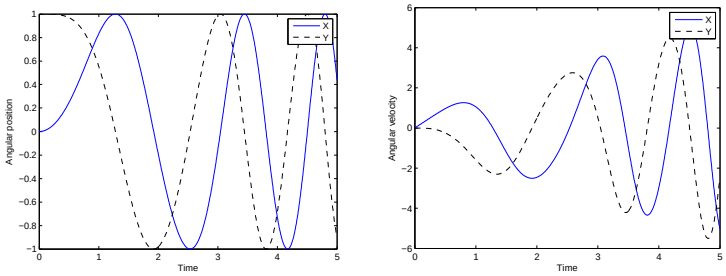


(g) Total Energy



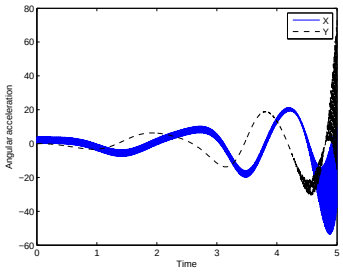
(h) Energy Change

Figure A.1: Circle Track System - Option 1 - U0(1,1,0)

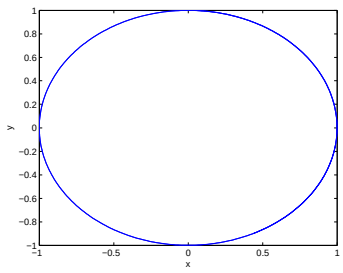


(a) Position

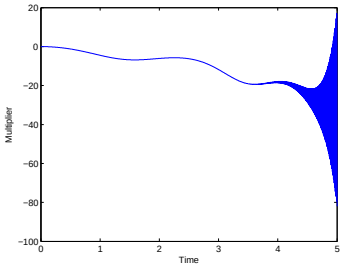
(b) Velocity



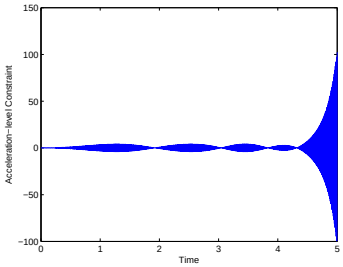
(c) Acceleration



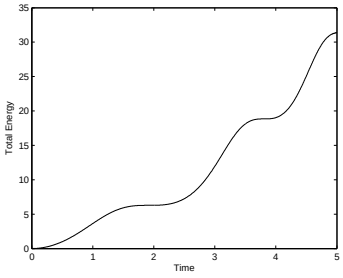
(d) Trajectory



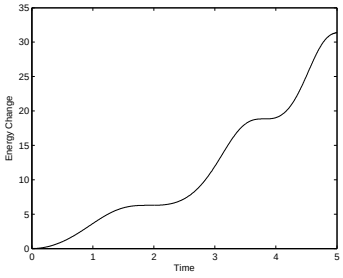
(e) Multiplier



(f) Acceleration Constraint

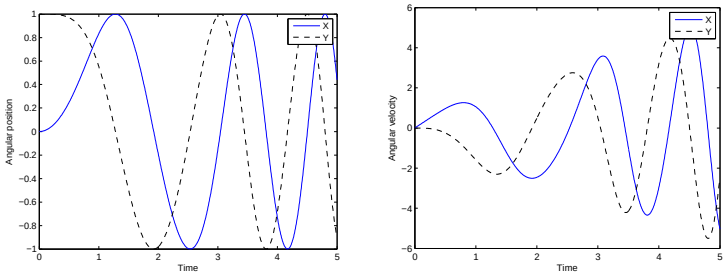


(g) Total Energy



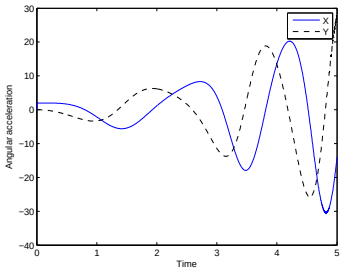
(h) Energy Change

Figure A.2: Circle Track System - Option 1 - V0(1,1,1)

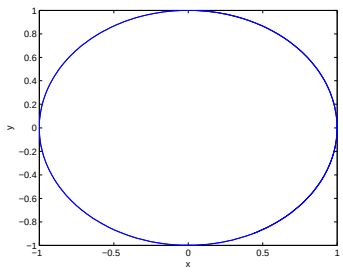


(a) Position

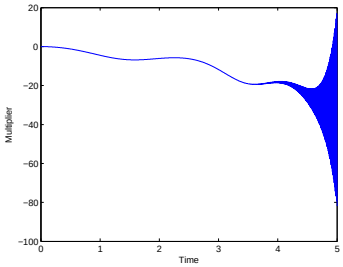
(b) Velocity



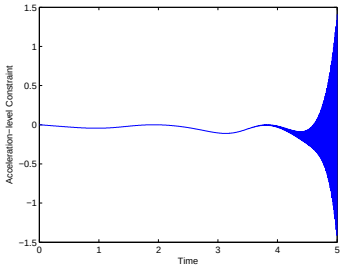
(c) Acceleration



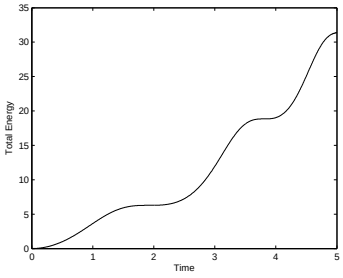
(d) Trajectory



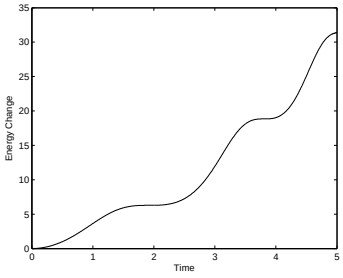
(e) Multiplier



(f) Acceleration Constraint

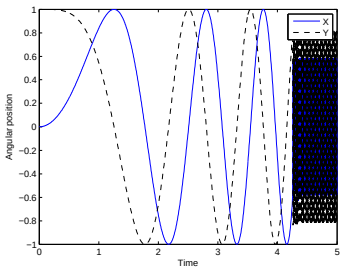


(g) Total Energy

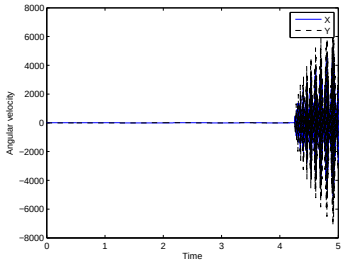


(h) Energy Change

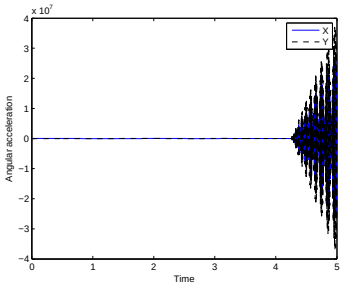
Figure A.3: Circle Track System - Option 1 - V0(1,1,0)



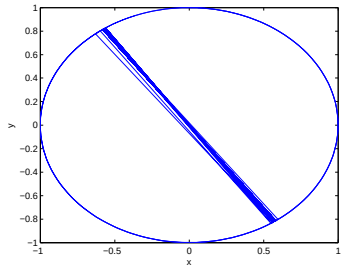
(a) Position



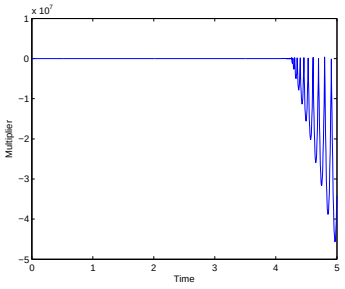
(b) Velocity



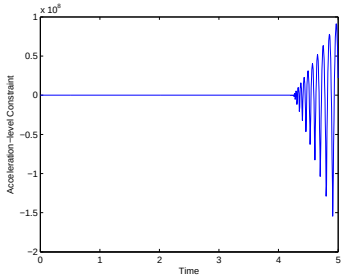
(c) Acceleration



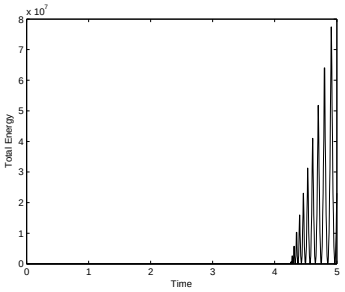
(d) Trajectory



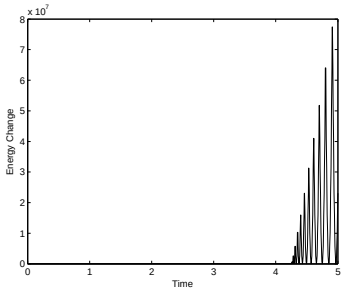
(e) Multiplier



(f) Acceleration Constraint

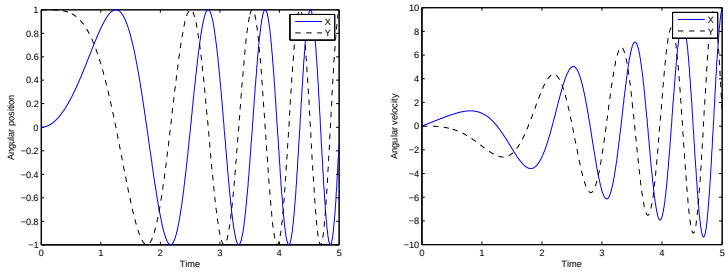


(g) Total Energy



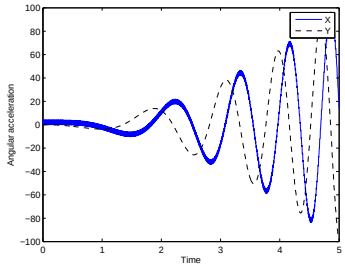
(h) Energy Change

Figure A.4: Circle Track System - Option 2 - U0(1,1,0)

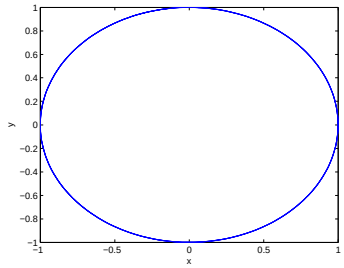


(a) Position

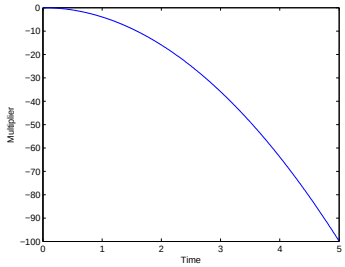
(b) Velocity



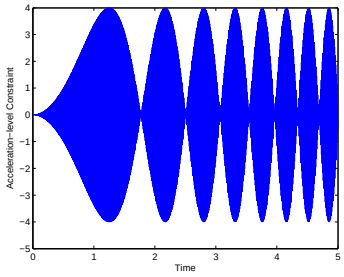
(c) Acceleration



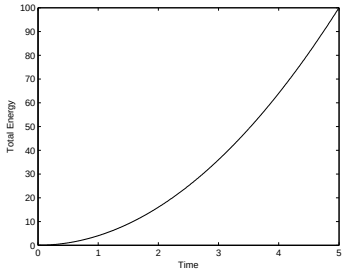
(d) Trajectory



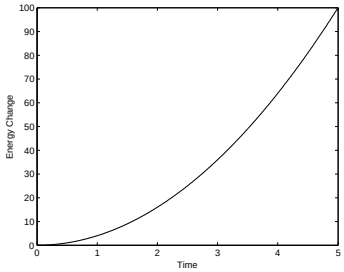
(e) Multiplier



(f) Acceleration Constraint

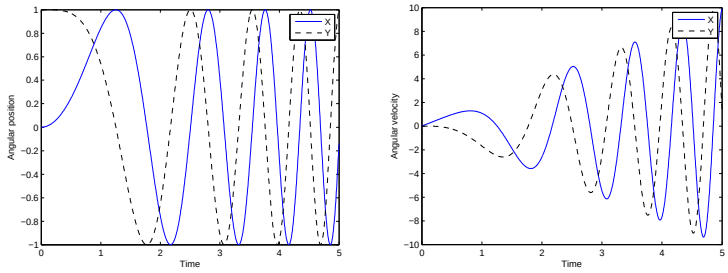


(g) Total Energy



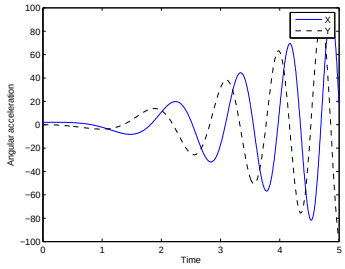
(h) Energy Change

Figure A.5: Circle Track System - Option 2 - V0(1,1,1)

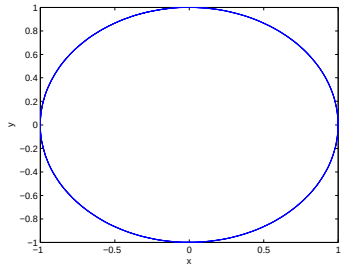


(a) Position

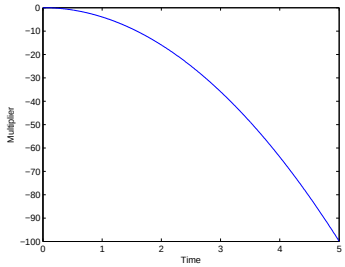
(b) Velocity



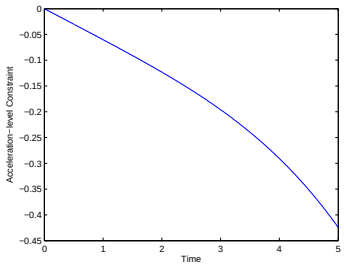
(c) Acceleration



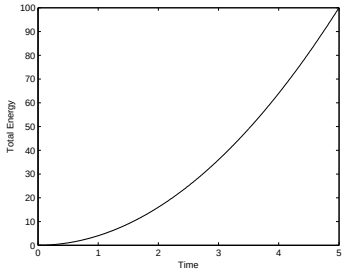
(d) Trajectory



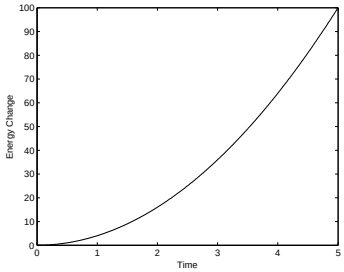
(e) Multiplier



(f) Acceleration Constraint

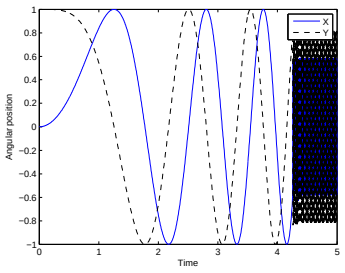


(g) Total Energy

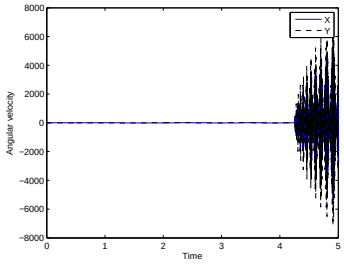


(h) Energy Change

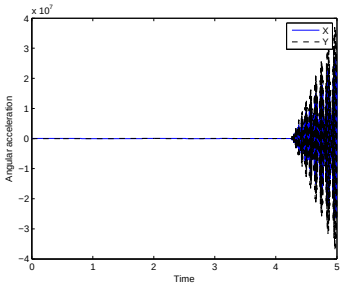
Figure A.6: Circle Track System - Option 2 - V0(1,1,0)



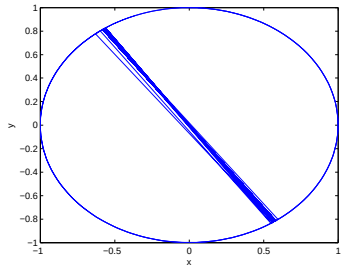
(a) Position



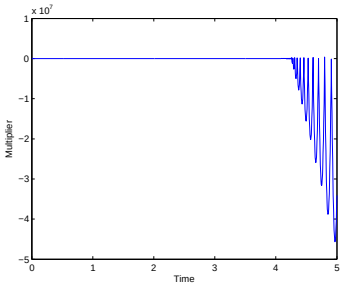
(b) Velocity



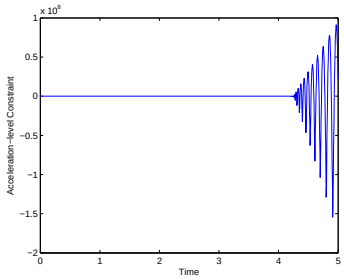
(c) Acceleration



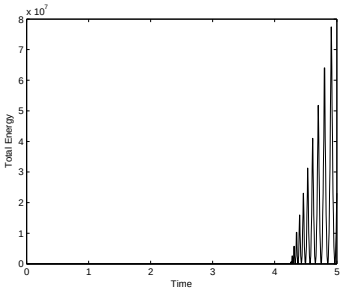
(d) Trajectory



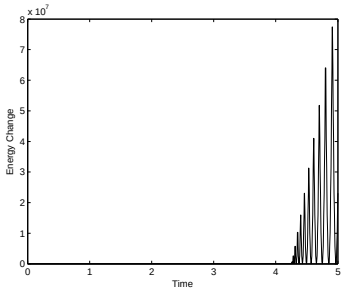
(e) Multiplier



(f) Acceleration Constraint

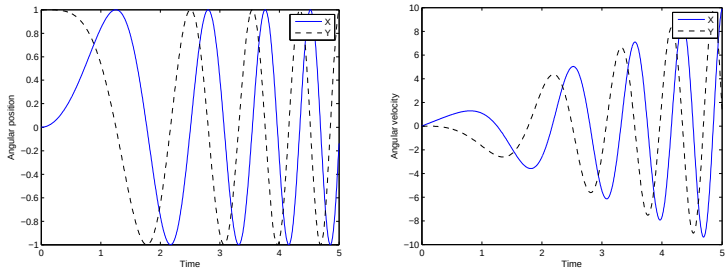


(g) Total Energy



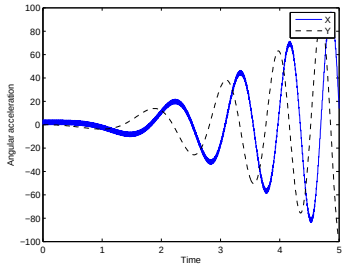
(h) Energy Change

Figure A.7: Circle Track System - Option 3 - U0(1,1,0)

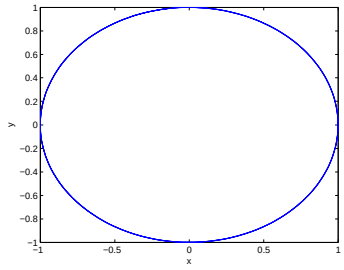


(a) Position

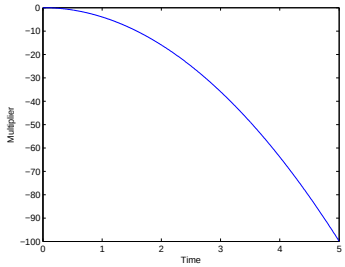
(b) Velocity



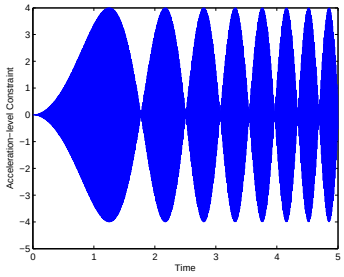
(c) Acceleration



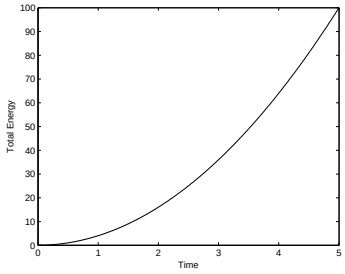
(d) Trajectory



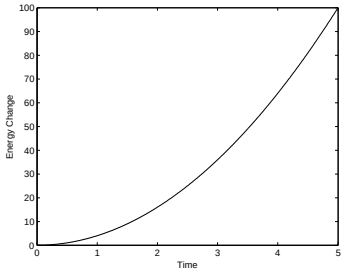
(e) Multiplier



(f) Acceleration Constraint

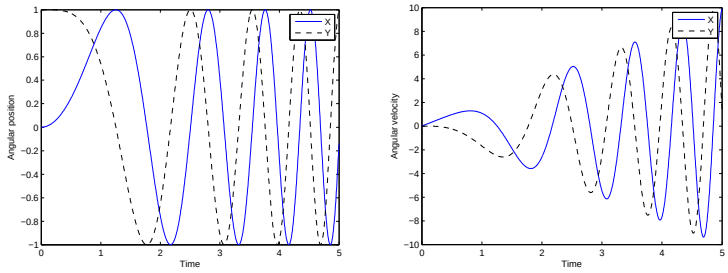


(g) Total Energy



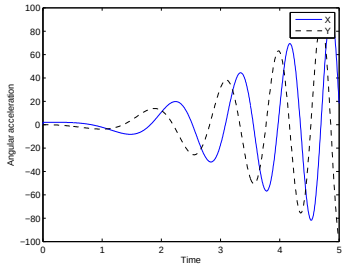
(h) Energy Change

Figure A.8: Circle Track System - Option 3 - V0(1,1,1)

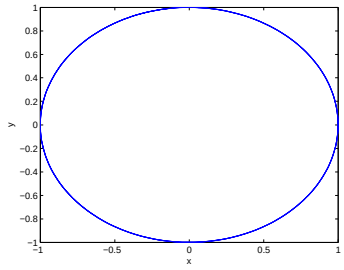


(a) Position

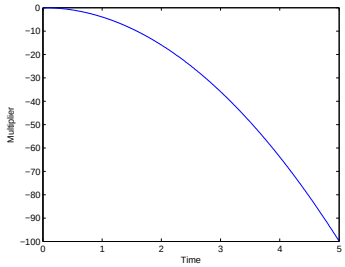
(b) Velocity



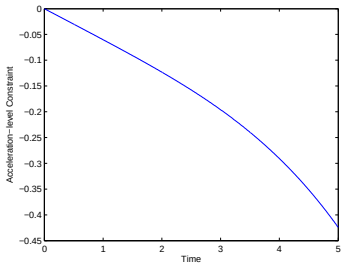
(c) Acceleration



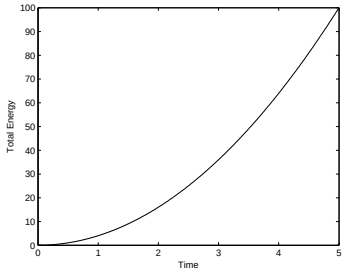
(d) Trajectory



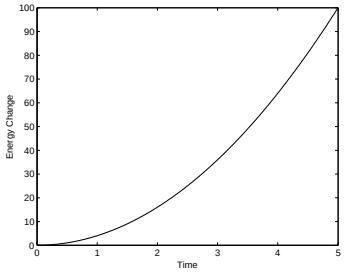
(e) Multiplier



(f) Acceleration Constraint

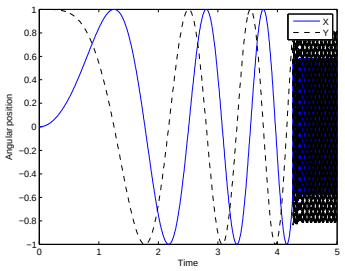


(g) Total Energy

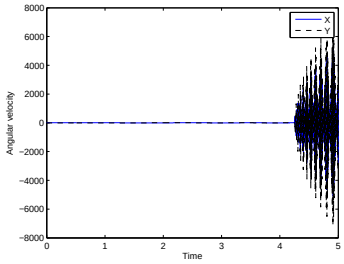


(h) Energy Change

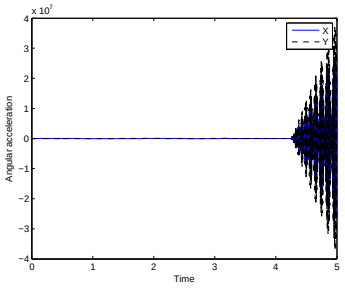
Figure A.9: Circle Track System - Option 3 - V0(1,1,0)



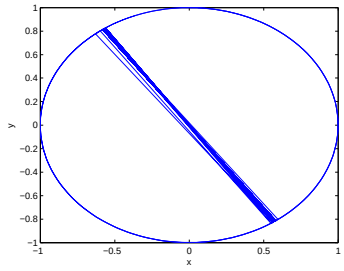
(a) Position



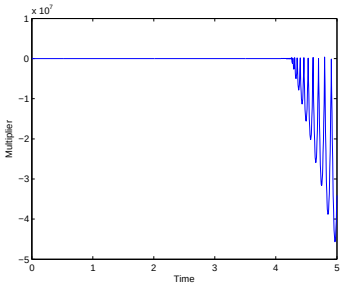
(b) Velocity



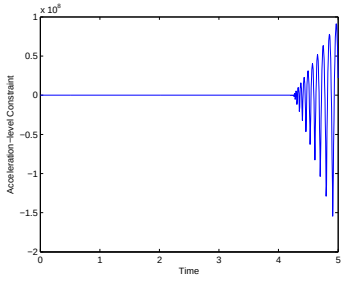
(c) Acceleration



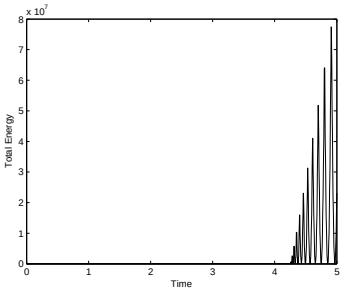
(d) Trajectory



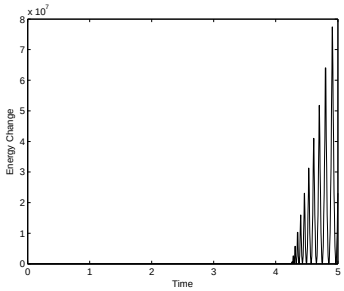
(e) Multiplier



(f) Acceleration Constraint

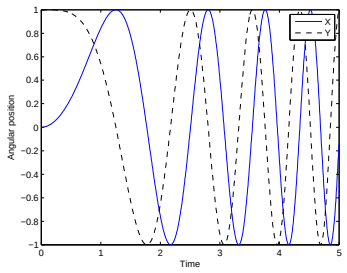


(g) Total Energy

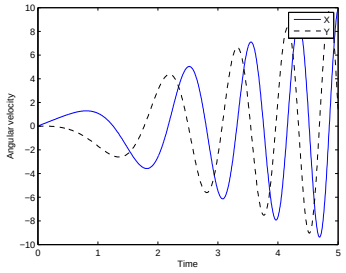


(h) Energy Change

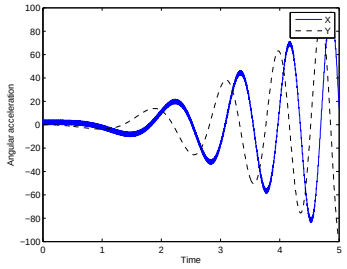
Figure A.10: Circle Track System - Option 4 - U0(1,1,0)



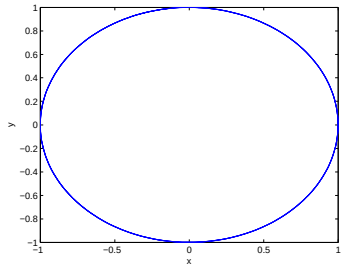
(a) Position



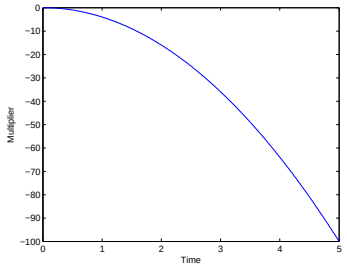
(b) Velocity



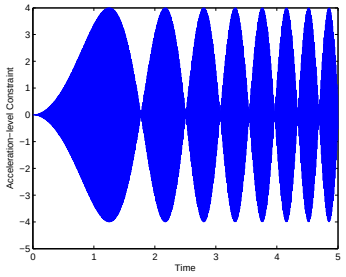
(c) Acceleration



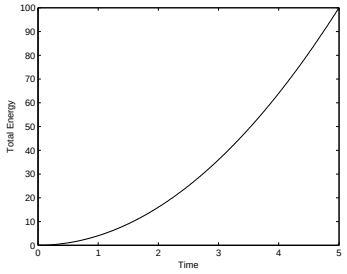
(d) Trajectory



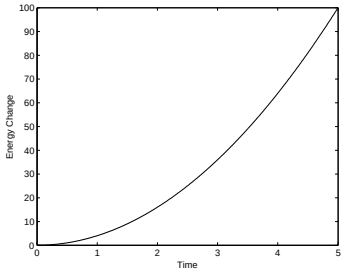
(e) Multiplier



(f) Acceleration Constraint

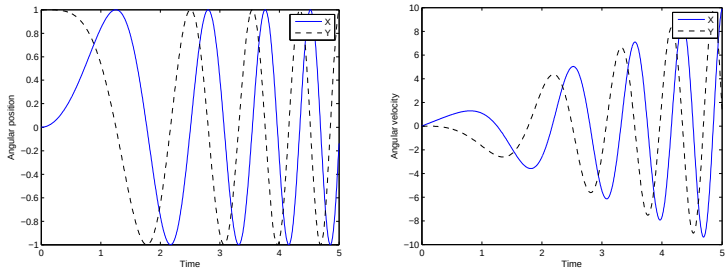


(g) Total Energy



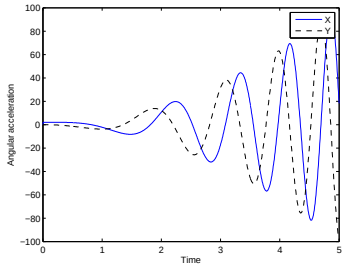
(h) Energy Change

Figure A.11: Circle Track System - Option 4 - V0(1,1,1)

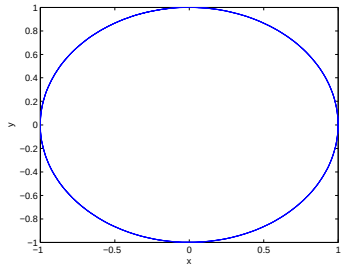


(a) Position

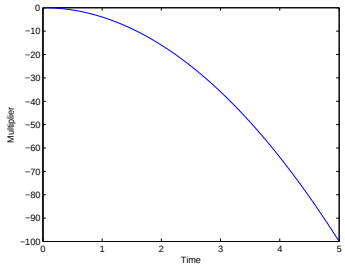
(b) Velocity



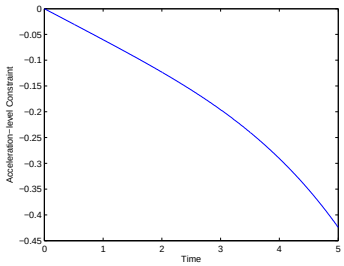
(c) Acceleration



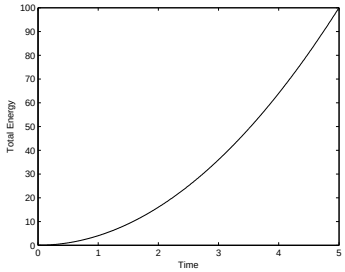
(d) Trajectory



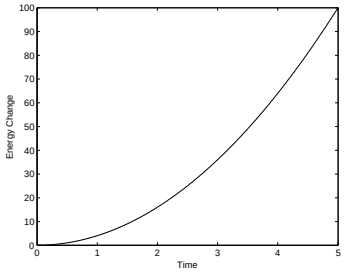
(e) Multiplier



(f) Acceleration Constraint

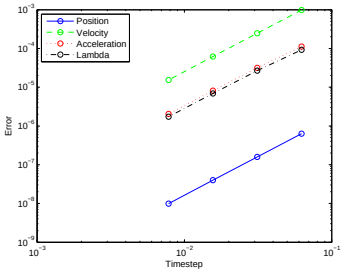


(g) Total Energy

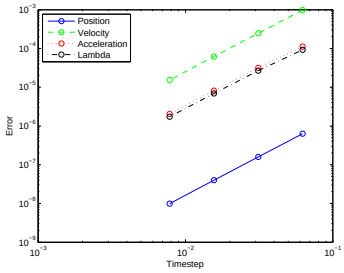


(h) Energy Change

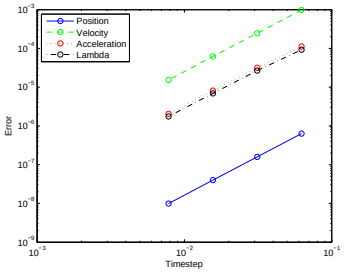
Figure A.12: Circle Track System - Option 4 - V0(1,1,0)



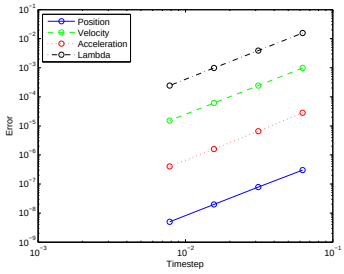
(a) Option 1 U0(1,1,0)



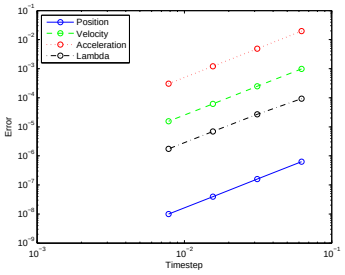
(b) Option 2 U0(1,1,0)



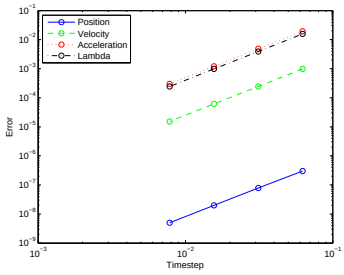
(c) Option 1 U0/V0(1,1,1)



(d) Option 2 U0/V0(1,1,1)

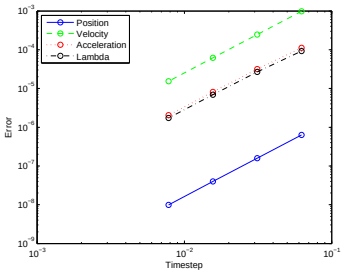


(e) Option 1 V0(1,1,0)

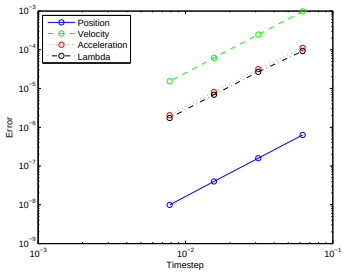


(f) Option 2 V0(1,1,0)

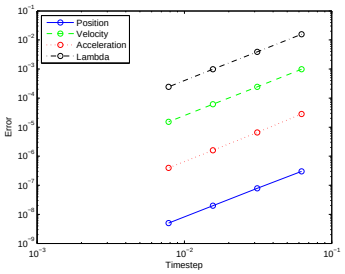
Figure A.13: Circular Track - Option 1, 2 Convergence



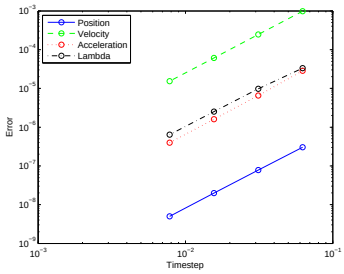
(a) Option 3 U0(1,1,0)



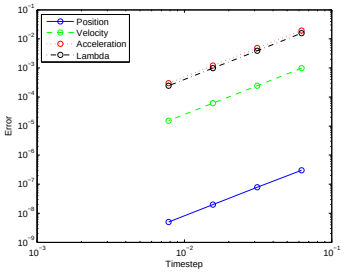
(b) Option 4 U0(1,1,0)



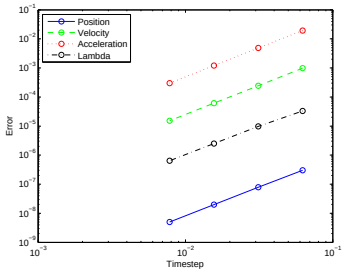
(c) Option 3 U0/V0(1,1,1)



(d) Option 4 U0/V0(1,1,1)



(e) Option 3 V0(1,1,0)

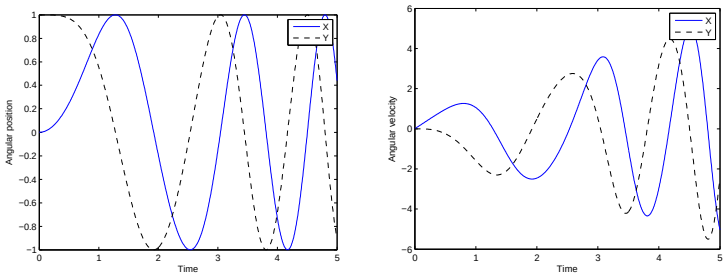


(f) Option 4 V0(1,1,0)

Figure A.14: Circle Track - Option 3, 4 Convergence

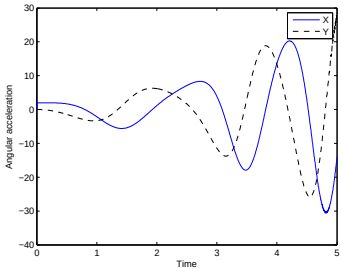
Two Field Form Results

The two field algorithm in its v-form representation (equivalent to V0(1,1,0) algorithm) results are shown in Figs. A.15-A.18. All simulation parameters are identical to the single field form simulations above.

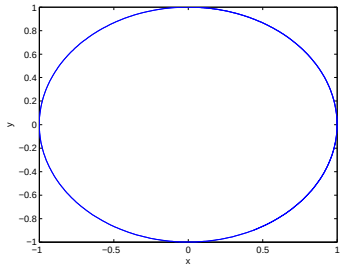


(a) Position

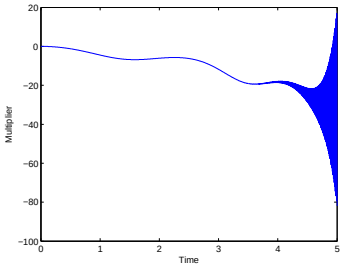
(b) Velocity



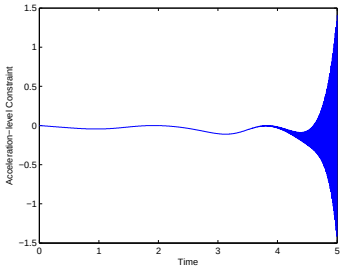
(c) Acceleration



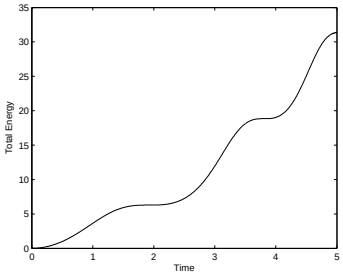
(d) Trajectory



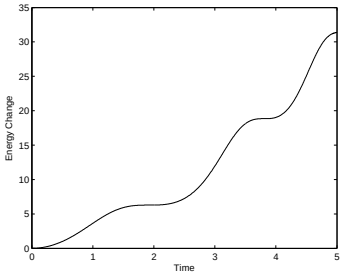
(e) Multiplier



(f) Acceleration Constraint

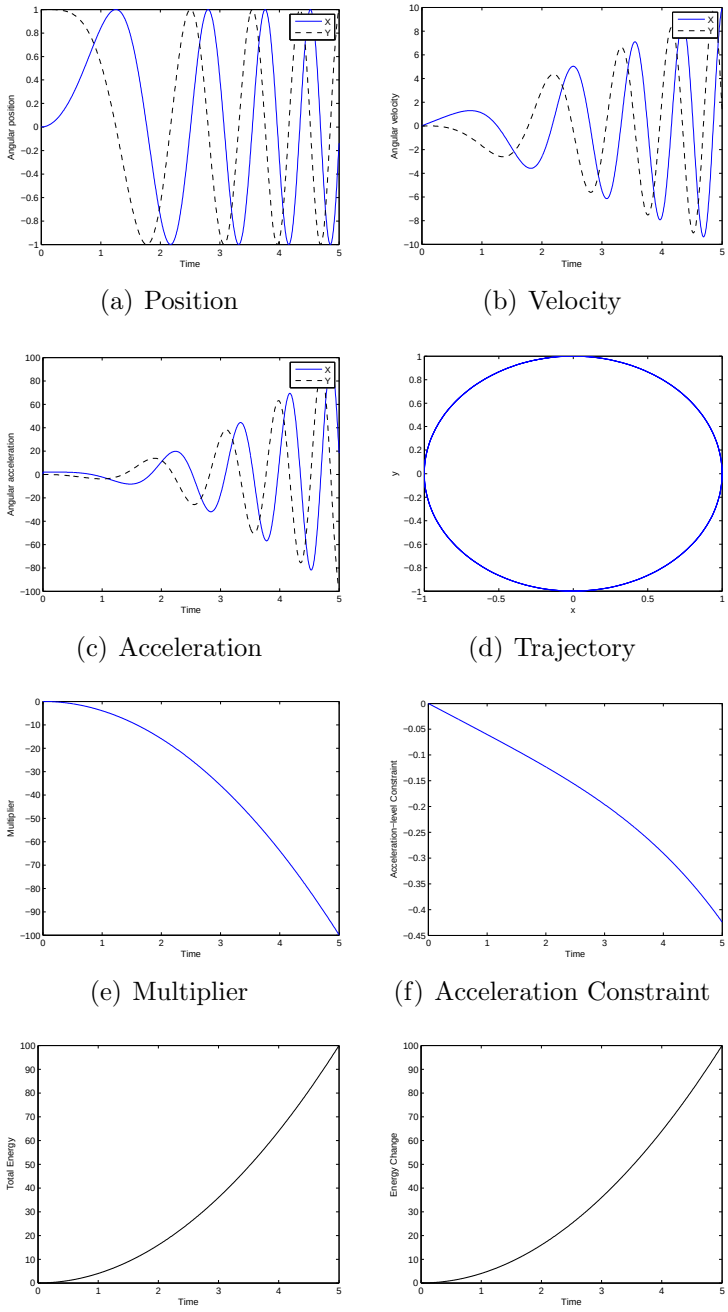


(g) Total Energy



(h) Energy Change

Figure A.15: Circle Track System - Option 1 - Two Field Form



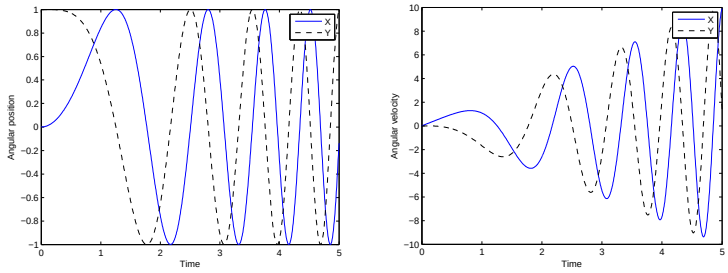
(a) Position (b) Velocity

(c) Acceleration (d) Trajectory

(e) Multiplier (f) Acceleration Constraint

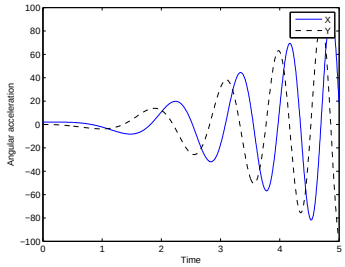
(g) Total Energy (h) Energy Change

Figure A.16: Circle Track System - Option 2 - Two Field Form

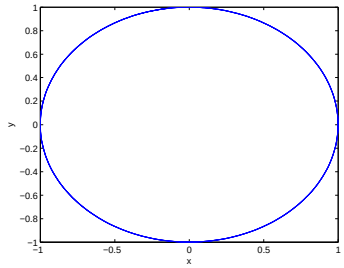


(a) Position

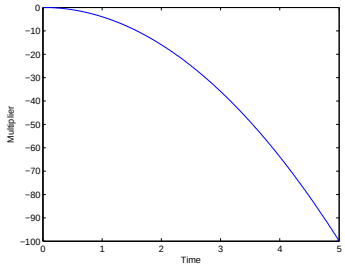
(b) Velocity



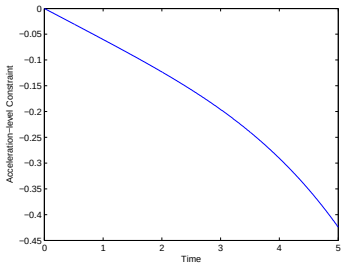
(c) Acceleration



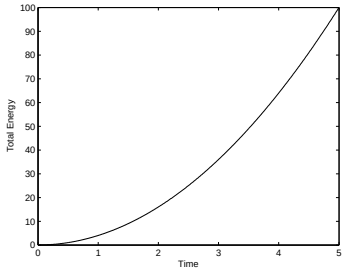
(d) Trajectory



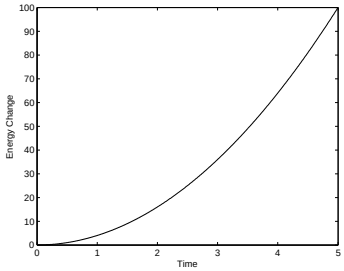
(e) Multiplier



(f) Acceleration Constraint

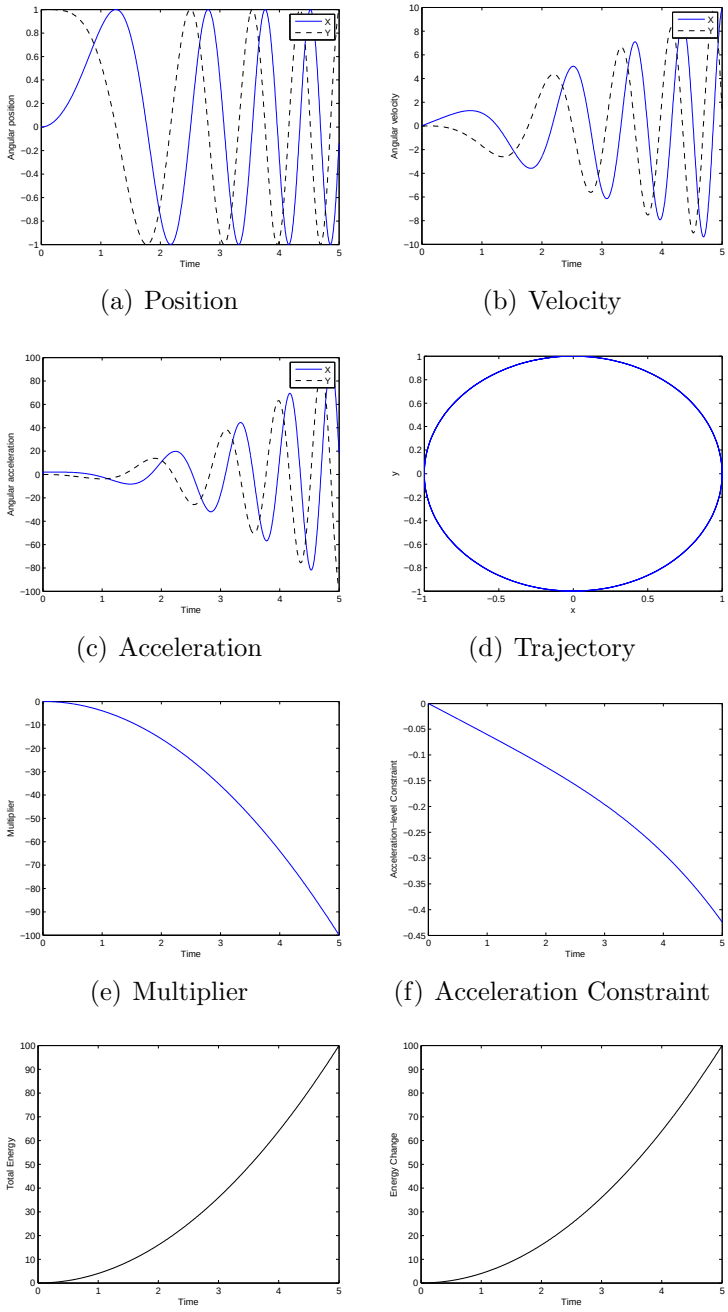


(g) Total Energy



(h) Energy Change

Figure A.17: Circle Track System - Option 3 - Two Field Form



(a) Position (b) Velocity

(c) Acceleration (d) Trajectory

(e) Multiplier (f) Acceleration Constraint

(g) Total Energy (h) Energy Change

Figure A.18: Circle Track System - Option 4 - Two Field Form

A.2 7 Bar Mechanism

The final illustrative example is the Andrew's 7 bar mechanism described in detail by Hairer and Wanner in [32]. Though this problem is not considered to represent a stiff set of equations (and thus, not a strong basis for presenting methods to overcome the numerical difficulties of DAEs) it is a very common example within the literature. The model consists of 7 rigid links and a single spring element. The layout of the system is shown in Fig. A.19.

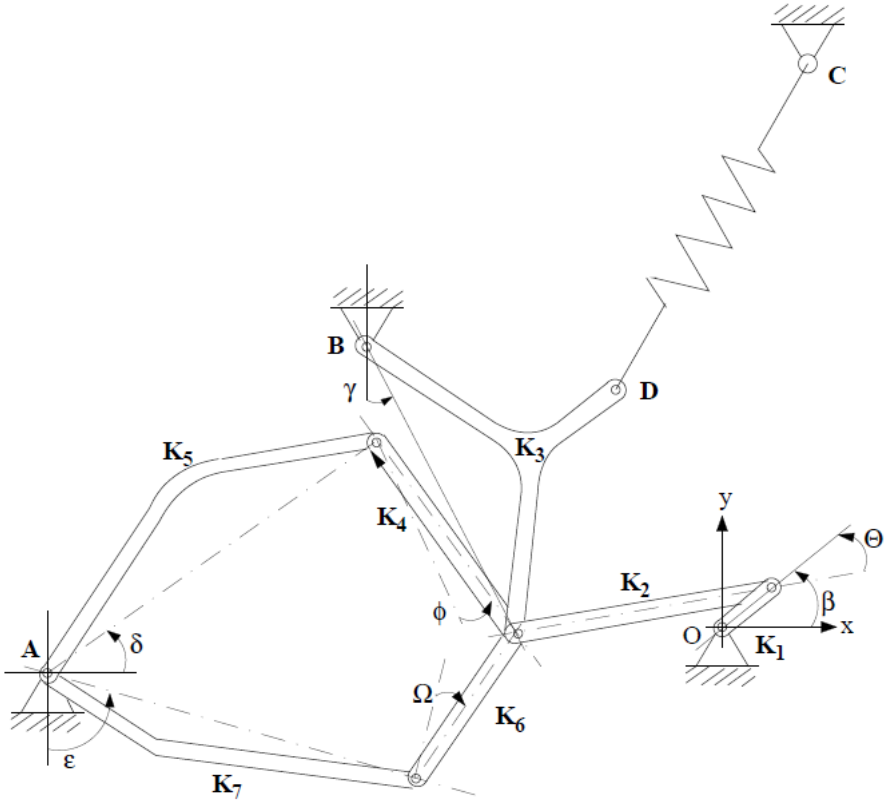


Figure A.19: 7 Bar Mechanism

The system can be described by the seven generalized coordinates

$$\mathbf{u} = [\beta, \theta, \gamma, \phi, \delta, \Omega, \epsilon]^T \tag{A.8}$$

but as suggested Negrut is done here using redundant coordinates. In addition to the angular orientation of each link the (x,y) coordinates of the center of mass of each link are also included in the system degrees of freedom leading to 21 overall degrees of freedom. This serves to simplify (make constant) the mass matrix as well as limit calls to inverse trig. functions. The coordinates are then of the form:

$$\mathbf{u} = [x_1, y_1, \beta, x_2, y_2, \theta, ..., x_7, y_7, \epsilon]^T \tag{A.9}$$

In general the problem is of the form:

$$\mathbf{M\ddot{u}} + \mathbf{B}^T(\mathbf{u})\boldsymbol{\lambda} = \mathbf{F} \tag{A.10}$$

$$\Phi(\mathbf{u}) = 0 \tag{A.11}$$

where the mass matrix is defined as the diagonal 21x21 matrix:

$$\mathbf{M} = diag([m_1, m_1, 0, m_2, m_2, 0, ...m_7, m_7, 0]) \tag{A.12}$$

and the nonzero elements of the (21x1) force vector are:

$$F(3,1) = T \tag{A.13}$$

$$F(7,1) = F_x \tag{A.14}$$

$$F(8,1) = F_y \tag{A.15}$$

$$F(9,1) = F_x(x_d - x_3) - F_y(y_d - y_3) \tag{A.16}$$

The constraint equations include the constraints on the Cartesian coordinates as well as the original six constraints from [32] as follows:

$$\Phi(\mathbf{u}) = \begin{bmatrix} x_1 - r_a\cos(\beta) \\ y_1 - r_a\sin(\beta) \\ x_2 - (r_r\cos(\beta) - da\cos(\theta)) \\ y_2 - (r_r\sin(\beta) - da\sin(\theta)) \\ x_3 - (x_b + s_a\sin(\gamma) + s_b\cos(\gamma)) \\ y_3 - (y_b - s_a\cos(\gamma) + s_b\sin(\gamma)) \\ x_4 - (x_a + z_t\cos(\delta) + (e - e_a)\sin(\phi)) \\ y_4 - (y_a + z_t\sin(\delta) - (e - e_a)\cos(\phi)) \\ x_5 - (x_a + t_a\cos(\delta) - t_b\sin(\delta)) \\ y_5 - (y_a + t_a\sin(\delta) + t_b\cos(\delta)) \\ x_6 - (x_a + u\sin(\epsilon) + (z_f - f_a)\cos(\Omega)) \\ y_6 - (y_a - u\cos(\epsilon) + (z_f - f_a)\sin(\Omega)) \\ x_7 - (x_a + u_a\sin(\epsilon) - u_b\cos(\epsilon)) \\ y_7 - (y_a - u_a\cos(\epsilon) - u_b\sin(\epsilon)) \\ r_r * \cos(\beta) - d\cos(\theta) - s_s\sin(\gamma) - x_b \\ r_r * \sin(\beta) - d\sin(\theta) + s_s\cos(\gamma) - y_b \\ r_r * \cos(\beta) - d\cos(\theta) - e\sin(\phi) - z_t\cos(\delta) - x_a \\ r_r * \sin(\beta) - d\sin(\theta) + e\cos(\phi) - z_t\sin(\delta) - y_a \\ r_r * \cos(\beta) - d\cos(\theta) - z_f\cos(\Omega) - u\sin(\epsilon) - x_a \\ r_r * \sin(\beta) - d\sin(\theta) - z_f\sin(\Omega) + u\cos(\epsilon) - y_a \end{bmatrix} \tag{A.17}$$

The physical constants of the problem are defined in Table A.1.

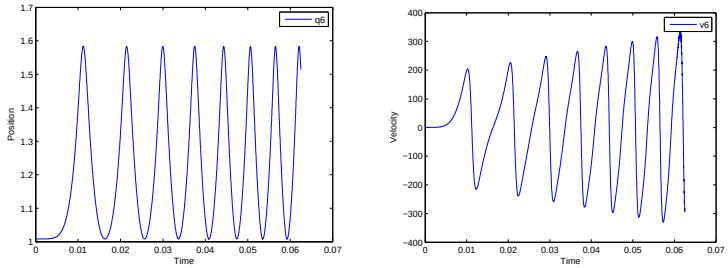
Figs. A.20-A.31 show the system response using a time step size of 2^{-15} for a duration of 2^{-4} seconds using the a-form representation of the single field form algorithm. The nonlinear iteration tolerance was set to 10^{-4}. It can be seen that for this non-stiff problem the U0/V0(1,1,1) and V0(1,1,0) algorithm using options 2, 3, and 4 all provide acceptable results. Analysis of CPU runtime and total nonlinear iteration counts, though, again identify the V0(1,1,0) - option 2 algorithm as being the optimal choice.

Finally, to ensure all schemes result in the expected second order accuracy in

$m_1 = 0.04325$	$r_r = 0.007$	$L = \sqrt{(x_d - x_c)^2 + (y_d - y_c)^2}$
$m_2 = 0.00365$	$r_a = 0.00092$	$F = -c_0(L - l_0)/L$
$m_3 = 0.02373$	$l_0 = 0.07785$	$F_x = F(x_d - x_c)$
$m_4 = 0.00706$	$c_0 = 4530$	$F_y = F(y_d - y_c)$
$m_5 = 0.07050$	$d = 0.028$	$t_a = 0.02308$
$m_6 = 0.00706$	$d_a = 0.0115$	$t_b = 0.009416$
$m_7 = 0.05498$	$e = 0.02$	$s_a = 0.01874$
$I_1 = 2.194 \times 10^{-6}$	$e_a = 0.01421$	$s_b = 0.01043$
$I_2 = 4.410 \times 10^{-7}$	$x_a = -0.06934$	$s_c = 0.018$
$I_3 = 5.255 \times 10^{-6}$	$y_a = -0.00227$	$s_d = 0.02$
$I_4 = 5.667 \times 10^{-7}$	$x_b = -0.03635$	$s_s = 0.035$
$I_5 = 1.169 \times 10^{-5}$	$y_b = 0.03273$	$u = 0.04$
$I_6 = 5.667 \times 10^{-7}$	$x_c = 0.014$	$u_a = 0.01228$
$I_7 = 1.912 \times 10^{-5}$	$y_c = 0.072$	$u_b = 0.00449$
$T = 0.033$	$x_d = s_d \cos\gamma + s_c \sin\gamma + x_b$	$z_f = 0.02$
$fa = 0.01421$	$yd = s_d \sin\gamma - s_c \cos\gamma + y_b$	$z_t = 0.04$

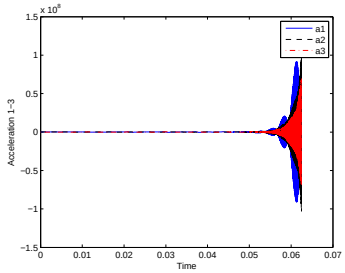
Table A.1: 7 Bar Mechanism Physical Constants

all variables, convergence plots are shown for each algorithm combination. Figs. A.32 and A.33 verify that all possible methods correctly maintain the expected accuracy.

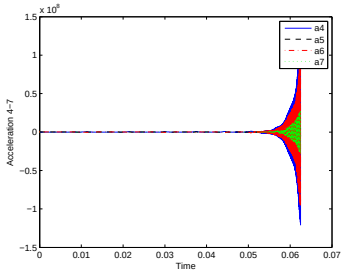


(a) Position

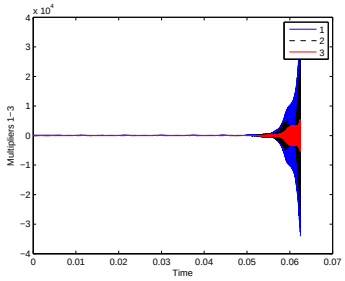
(b) Velocity



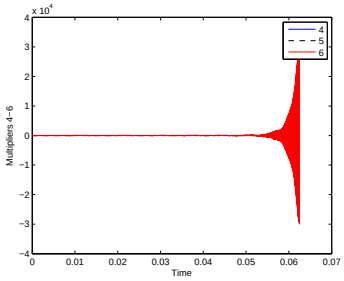
(c) Acceleration



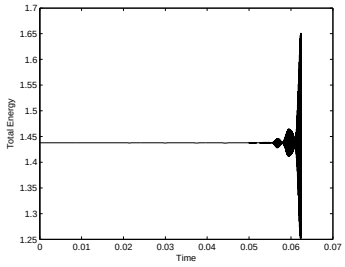
(d) Acceleration cont.



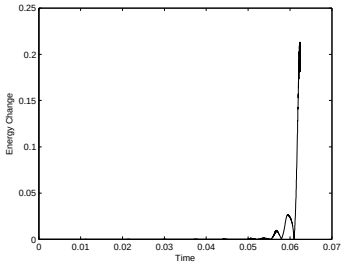
(e) Multipliers



(f) Multipliers cont.

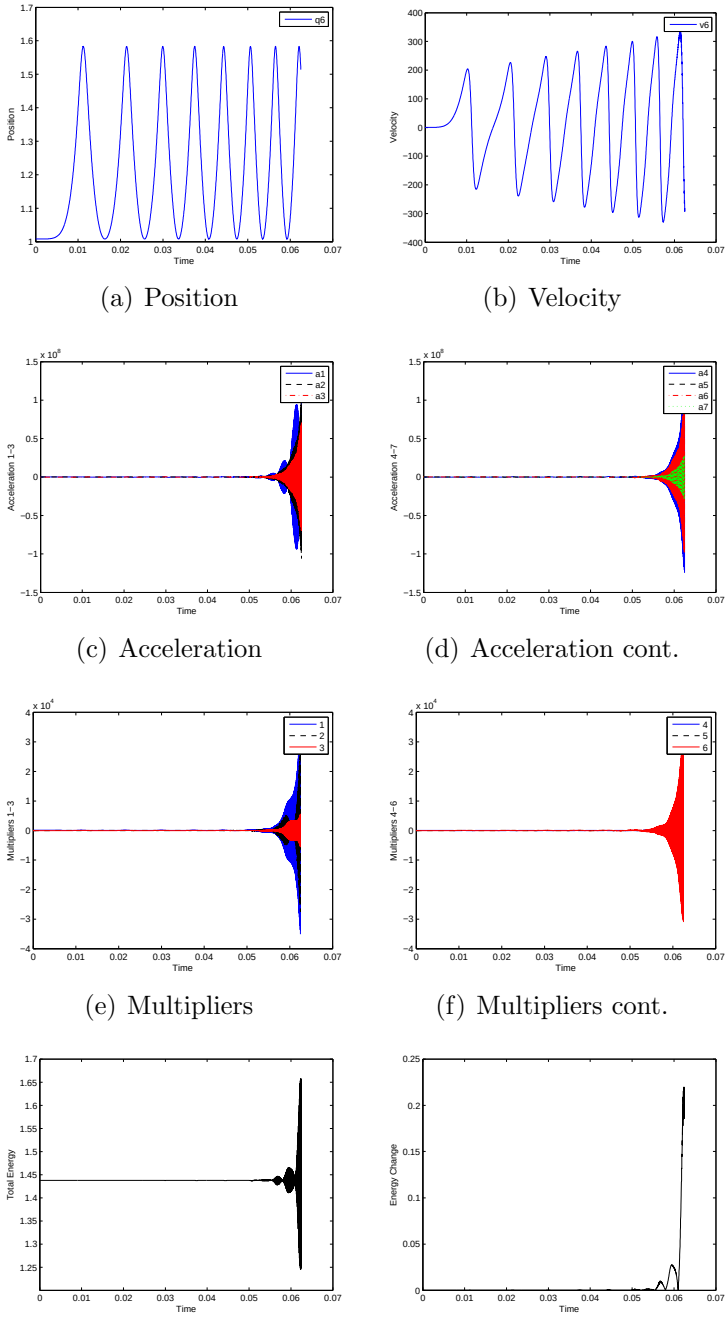


(g) Total Energy



(h) Energy Change

Figure A.20: 7 Bar Mechanism - Option 1 - U0(1,1,0)



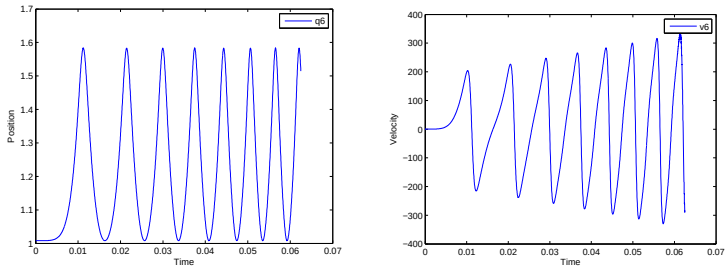
(a) Position (b) Velocity

(c) Acceleration (d) Acceleration cont.

(e) Multipliers (f) Multipliers cont.

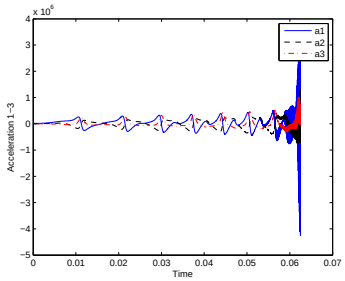
(g) Total Energy (h) Energy Change

Figure A.21: 7 Bar Mechanism - Option 1 - V0(1,1,1)

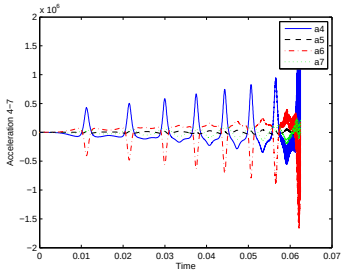


(a) Position

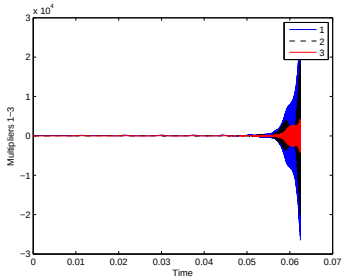
(b) Velocity



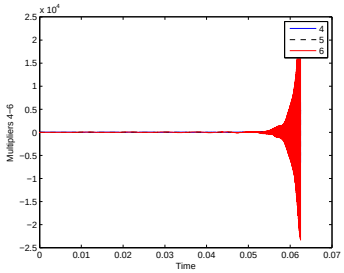
(c) Acceleration



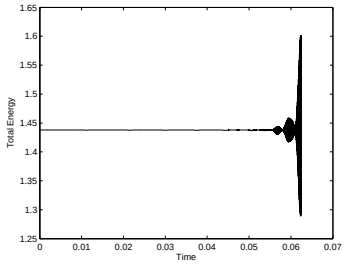
(d) Acceleration cont.



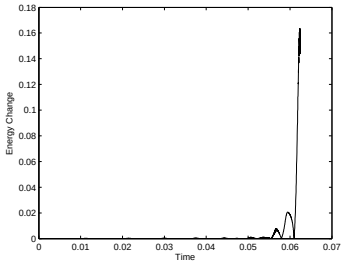
(e) Multipliers



(f) Multipliers cont.

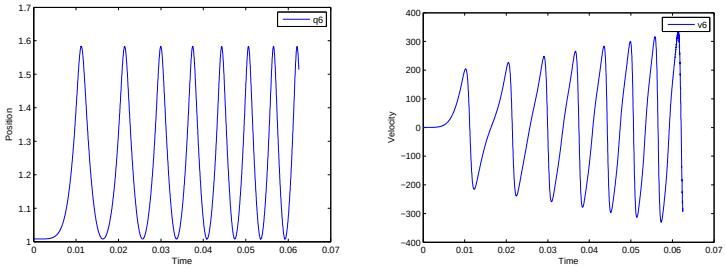


(g) Total Energy



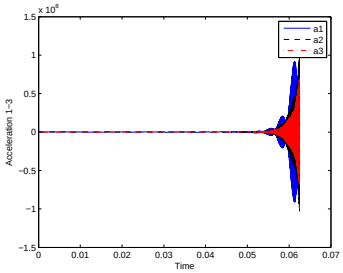
(h) Energy Change

Figure A.22: 7 Bar Mechanism - Option 1 - V0(1,1,0)

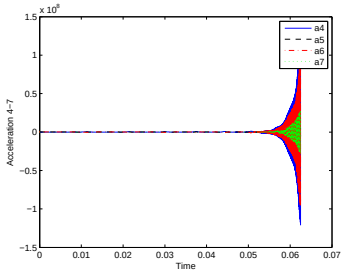


(a) Position

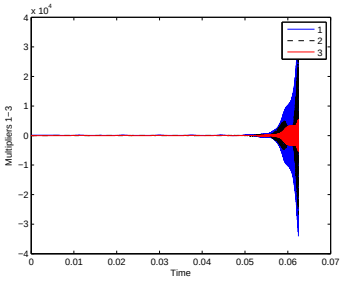
(b) Velocity



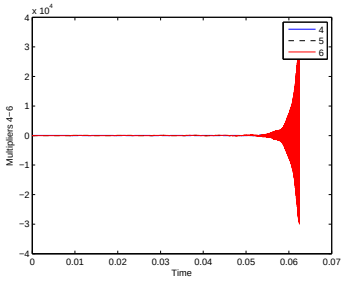
(c) Acceleration



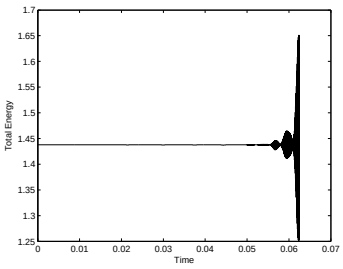
(d) Acceleration cont.



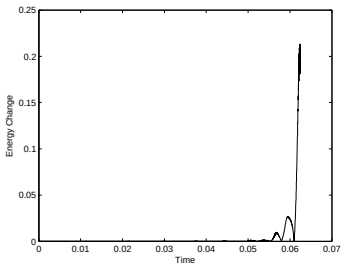
(e) Multipliers



(f) Multipliers cont.

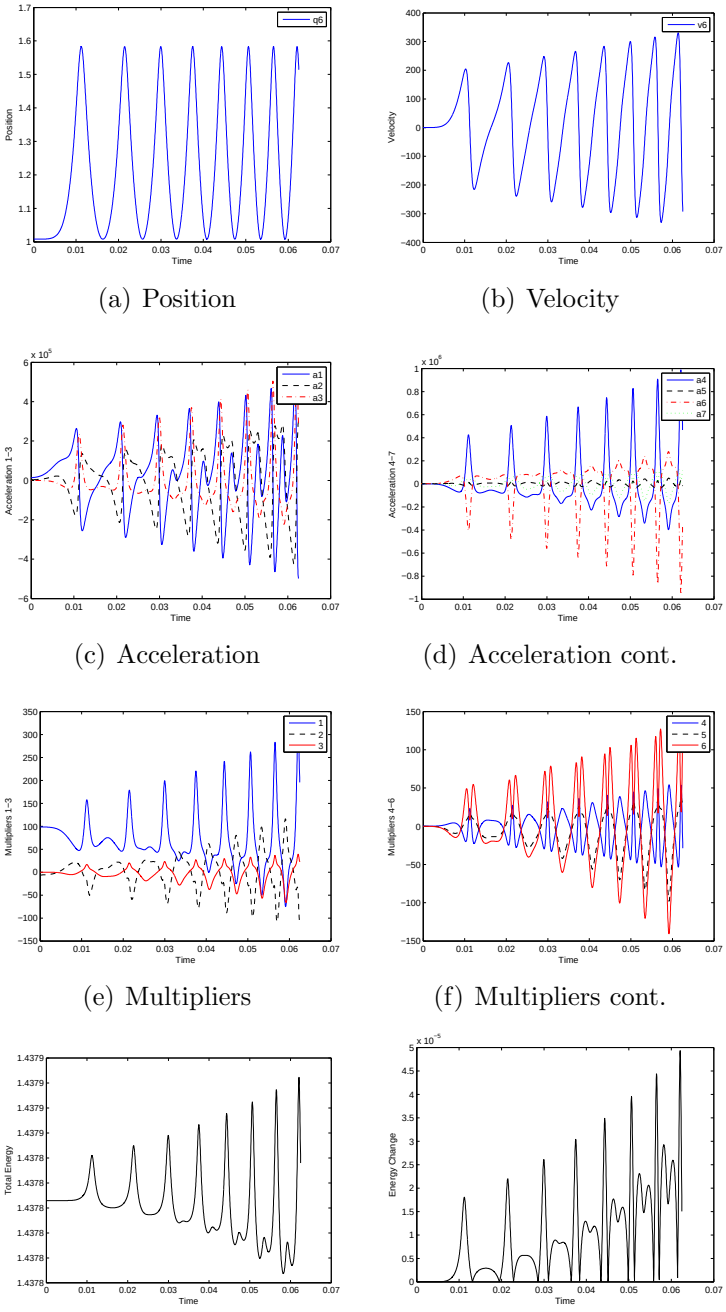


(g) Total Energy



(h) Energy Change

Figure A.23: 7 Bar Mechanism - Option 2 - U0(1,1,0)



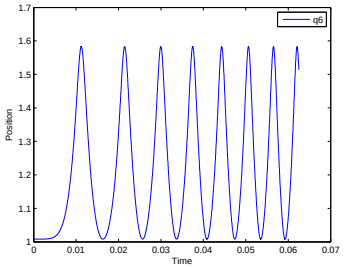
(a) Position (b) Velocity

(c) Acceleration (d) Acceleration cont.

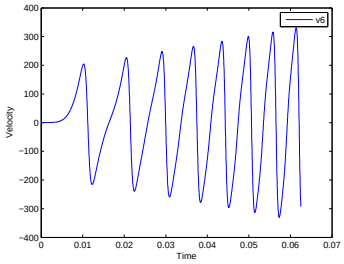
(e) Multipliers (f) Multipliers cont.

(g) Total Energy (h) Energy Change

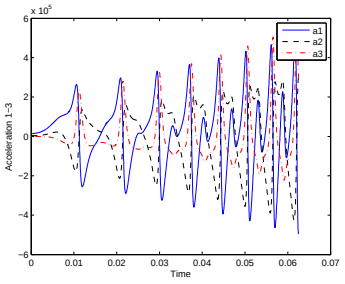
Figure A.24: 7 Bar Mechanism - Option 2 - V0(1,1,1)



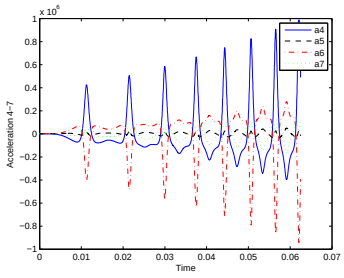
(a) Position



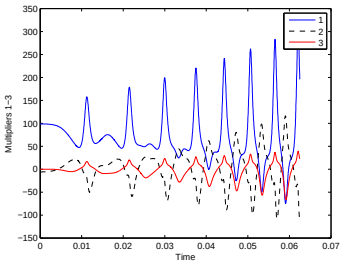
(b) Velocity



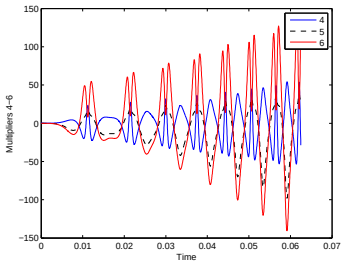
(c) Acceleration



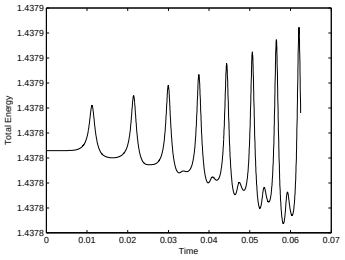
(d) Acceleration cont.



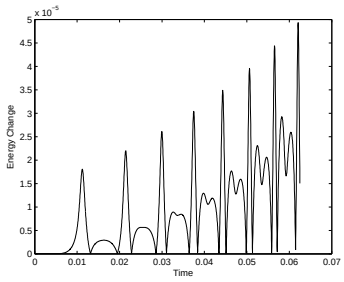
(e) Multipliers



(f) Multipliers cont.

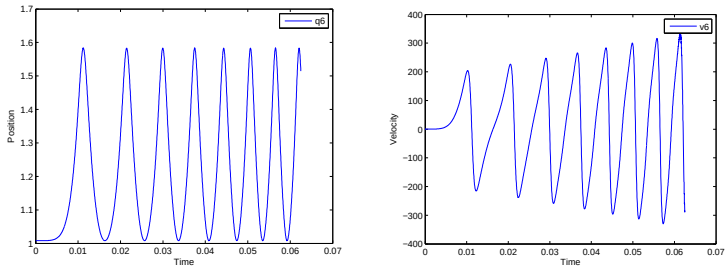


(g) Total Energy



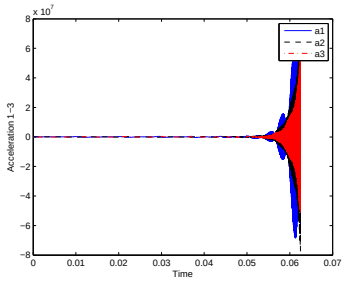
(h) Energy Change

Figure A.25: 7 Bar Mechanism - Option 2 - V0(1,1,0)

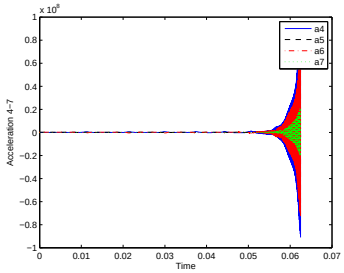


(a) Position

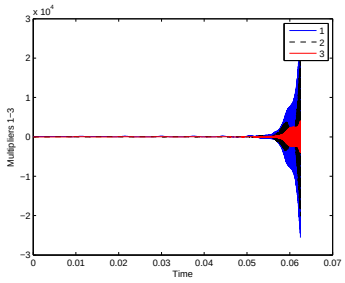
(b) Velocity



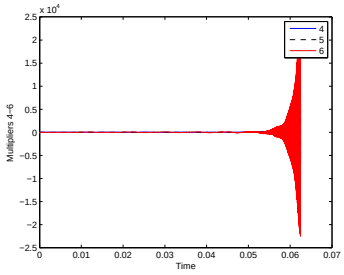
(c) Acceleration



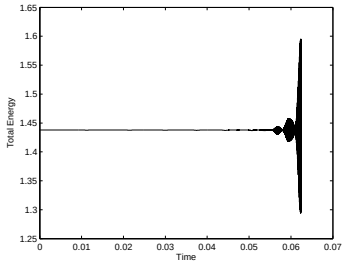
(d) Acceleration cont.



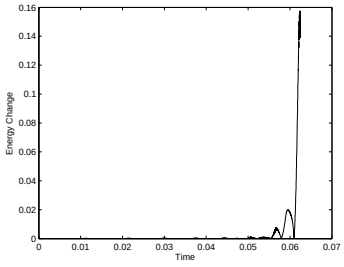
(e) Multipliers



(f) Multipliers cont.

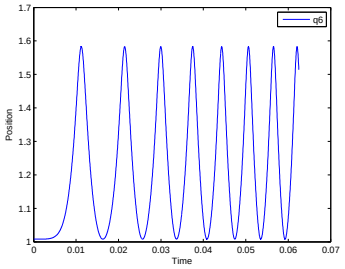


(g) Total Energy

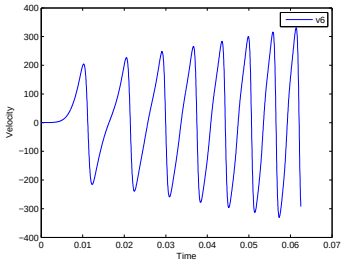


(h) Energy Change

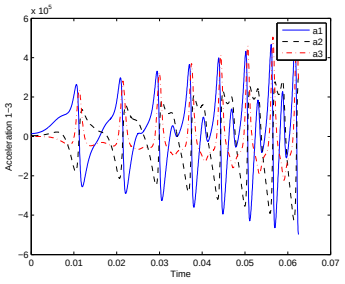
Figure A.26: 7 Bar Mechanism - Option 3 - U0(1,1,0)



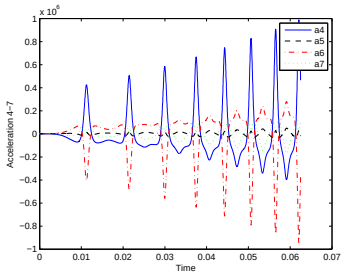
(a) Position



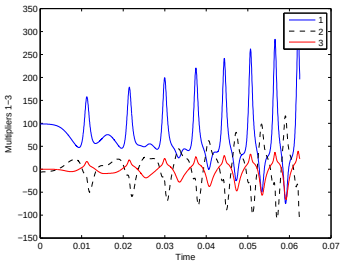
(b) Velocity



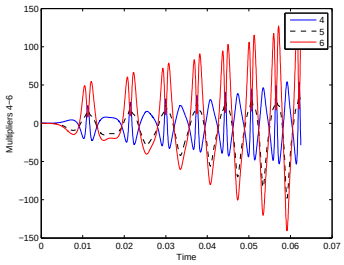
(c) Acceleration



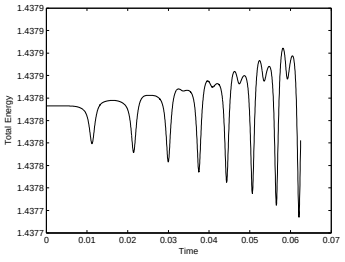
(d) Acceleration cont.



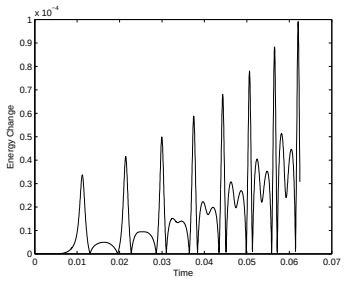
(e) Multipliers



(f) Multipliers cont.

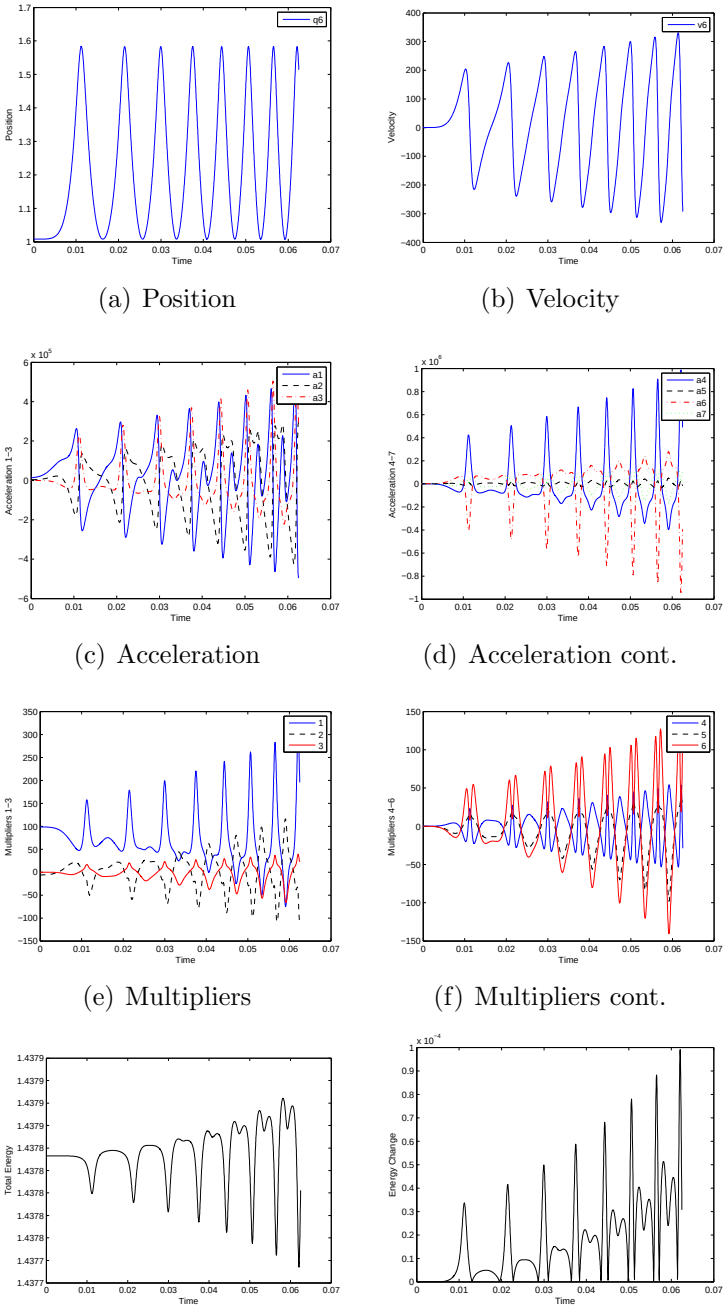


(g) Total Energy



(h) Energy Change

Figure A.27: 7 Bar Mechanism - Option 3 - V0(1,1,1)



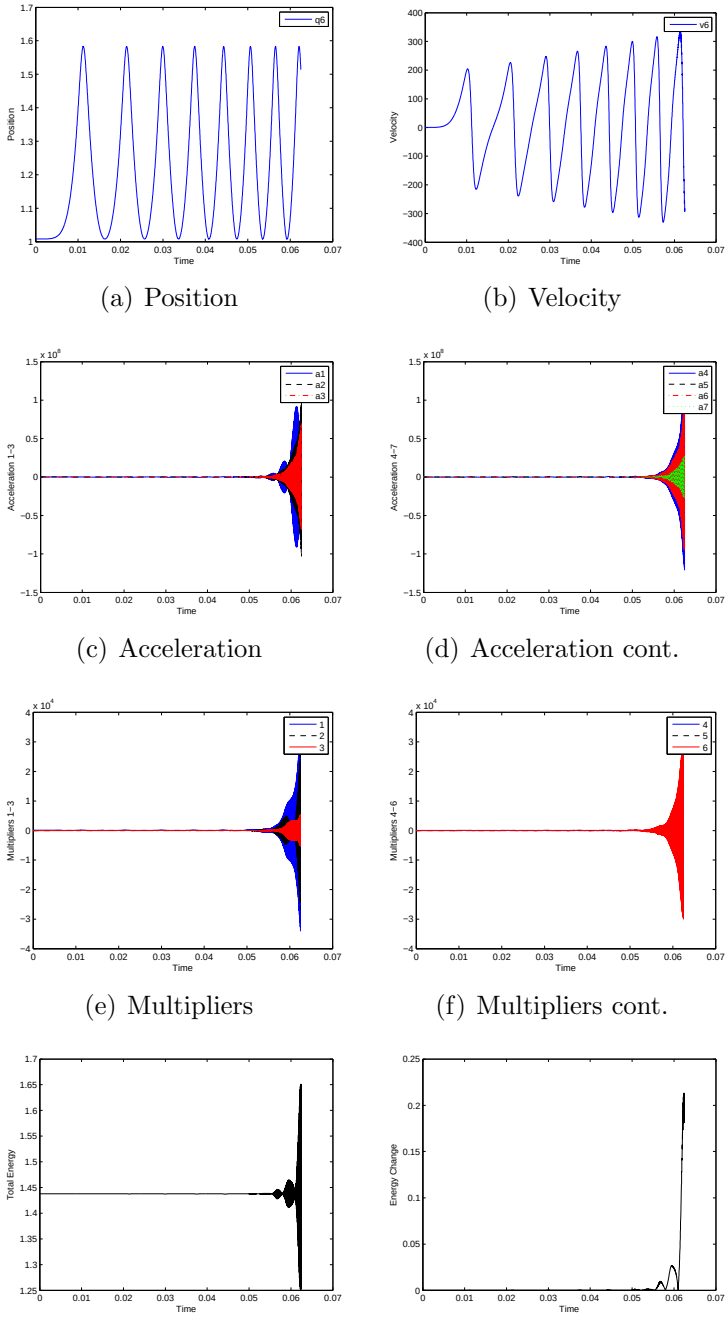
(a) Position (b) Velocity

(c) Acceleration (d) Acceleration cont.

(e) Multipliers (f) Multipliers cont.

(g) Total Energy (h) Energy Change

Figure A.28: 7 Bar Mechanism - Option 3 - V0(1,1,0)



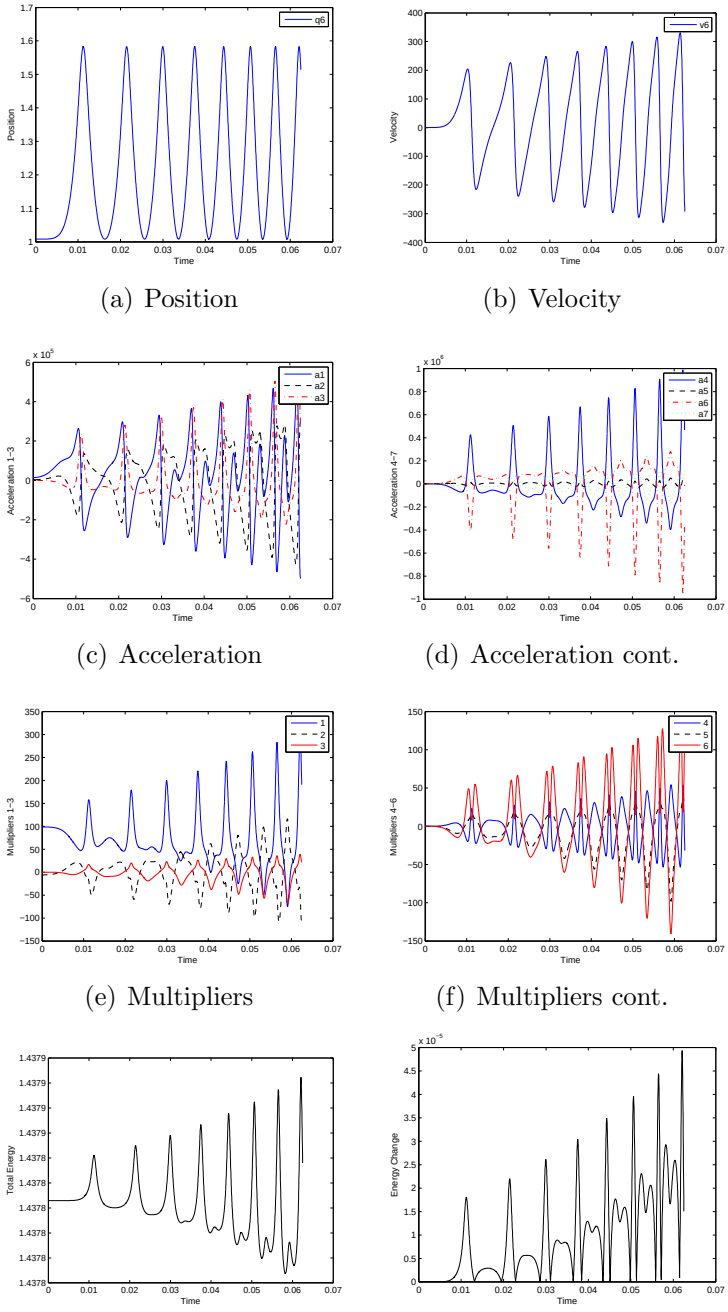
(a) Position (b) Velocity

(c) Acceleration (d) Acceleration cont.

(e) Multipliers (f) Multipliers cont.

(g) Total Energy (h) Energy Change

Figure A.29: 7 Bar Mechanism - Option 4 - U0(1,1,0)



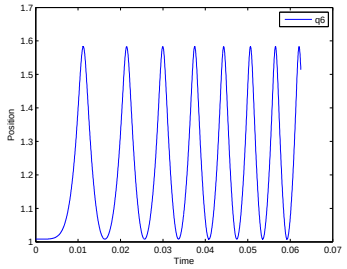
(a) Position (b) Velocity

(c) Acceleration (d) Acceleration cont.

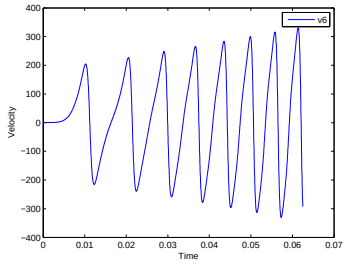
(e) Multipliers (f) Multipliers cont.

(g) Total Energy (h) Energy Change

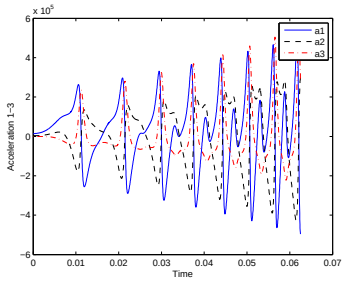
Figure A.30: 7 Bar Mechanism - Option 4 - V0(1,1,1)



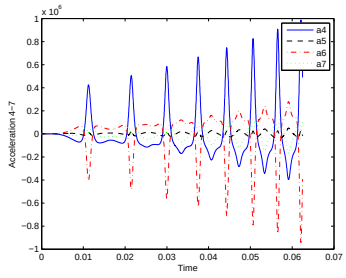
(a) Position



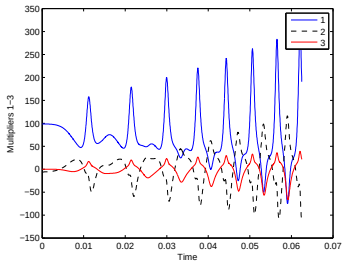
(b) Velocity



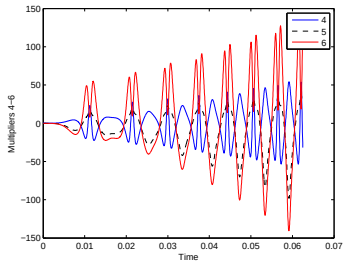
(c) Acceleration



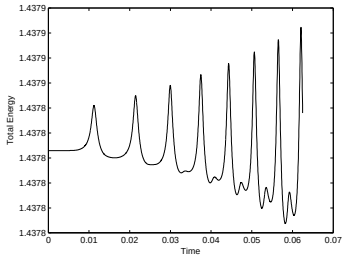
(d) Acceleration cont.



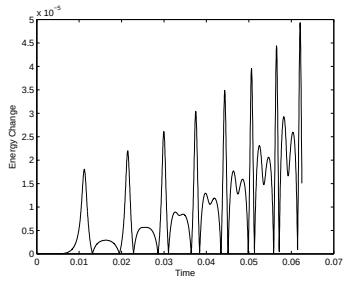
(e) Multipliers



(f) Multipliers cont.

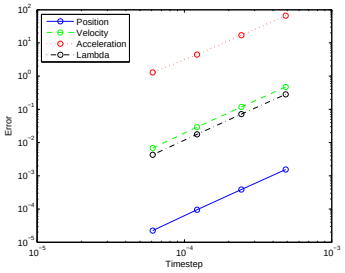


(g) Total Energy

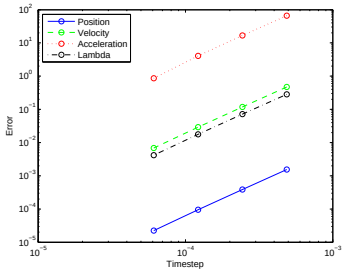


(h) Energy Change

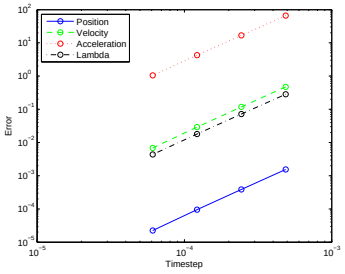
Figure A.31: 7 Bar Mechanism - Option 4 - V0(1,1,0)



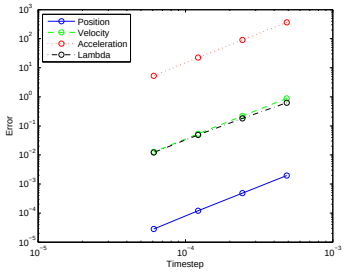
(a) Option 1 U0(1,1,0)



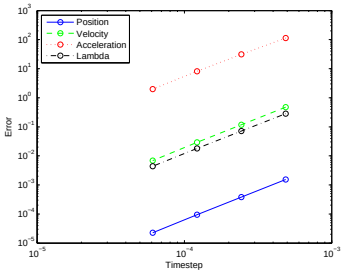
(b) Option 2 U0(1,1,0)



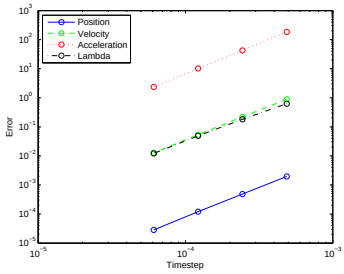
(c) Option 1 U0/V0(1,1,1)



(d) Option 2 U0/V0(1,1,1)

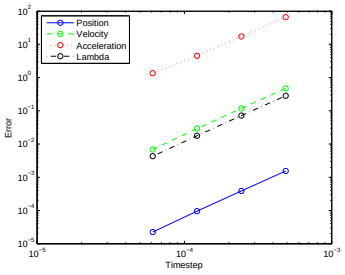


(e) Option 1 V0(1,1,0)

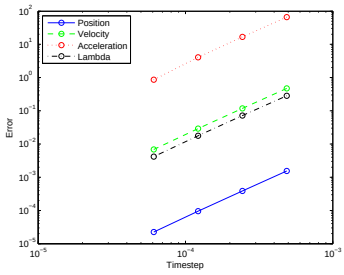


(f) Option 2 V0(1,1,0)

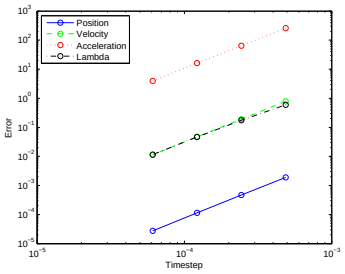
Figure A.32: 7 Bar - Option 1, 2 Convergence



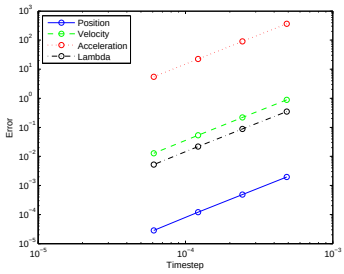
(a) Option 3 U0(1,1,0)



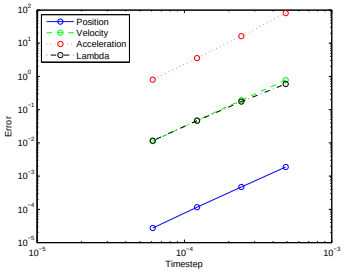
(b) Option 4 U0(1,1,0)



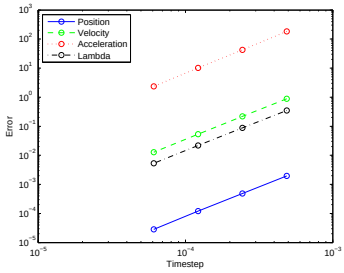
(c) Option 3 U0/V0(1,1,1)



(d) Option 4 U0/V0(1,1,1)



(e) Option 3 V0(1,1,0)

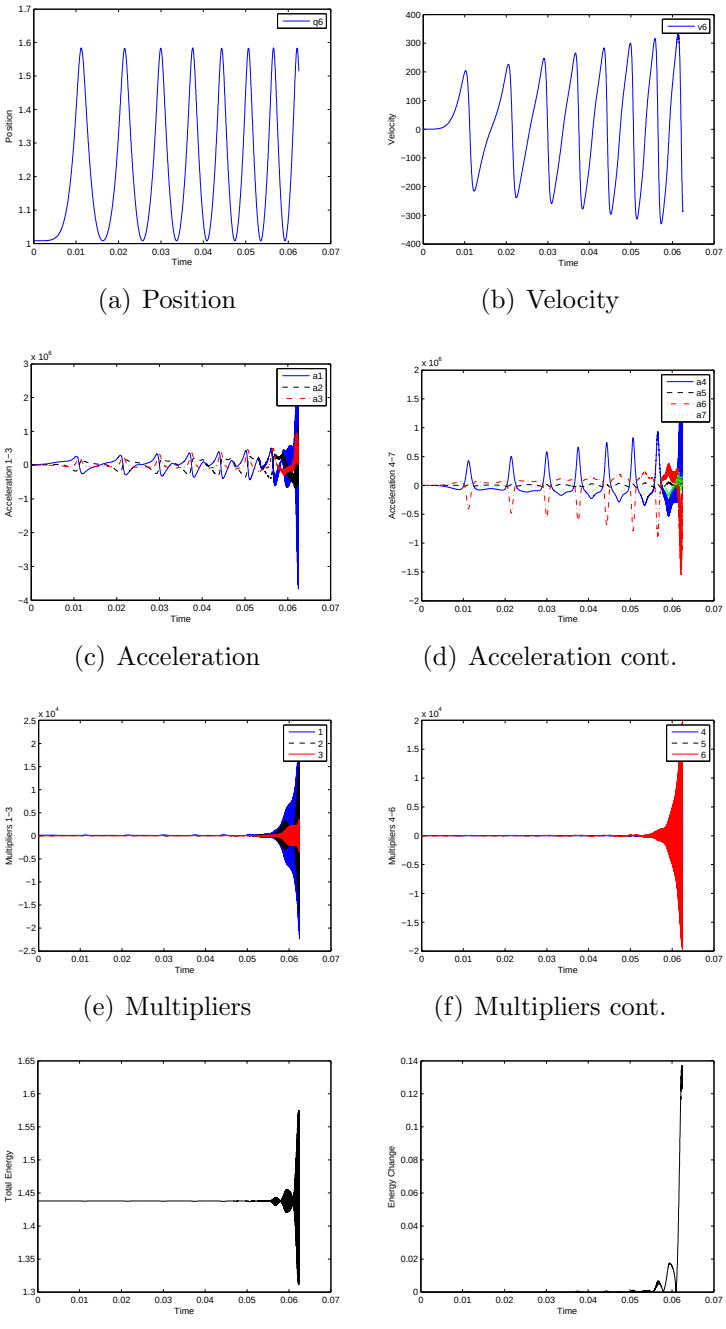


(f) Option 4 V0(1,1,0)

Figure A.33: 7 Bar - Option 3, 4 Convergence

Two Field Form Results

The two field algorithm in its v-form representation (equivalent to V0(1,1,0) algorithm) results are shown in Figs. A.34-A.37. All simulation parameters are identical to the single field form simulations above.



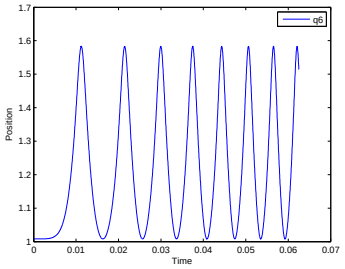
(a) Position (b) Velocity

(c) Acceleration (d) Acceleration cont.

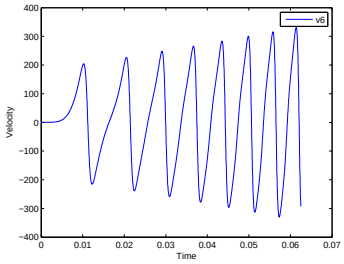
(e) Multipliers (f) Multipliers cont.

(g) Total Energy (h) Energy Change

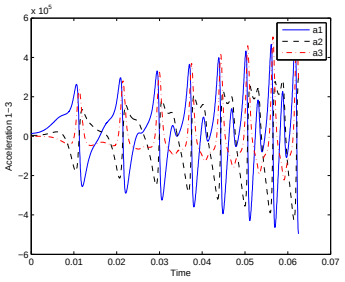
Figure A.34: 7 Bar Mechanism - Option 1 - Two Field Form



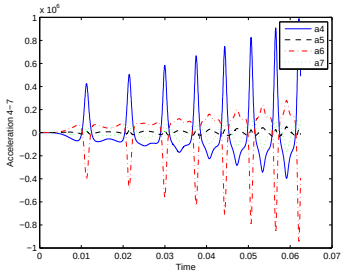
(a) Position



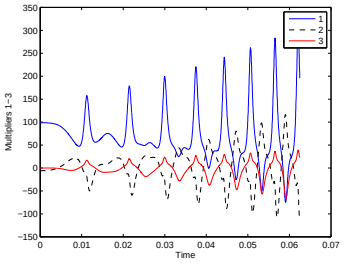
(b) Velocity



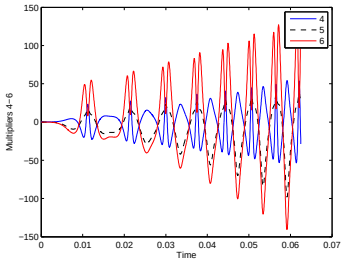
(c) Acceleration



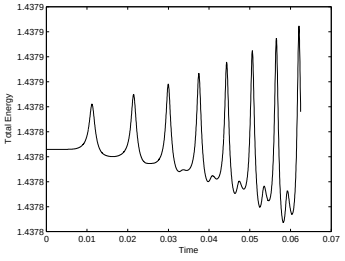
(d) Acceleration cont.



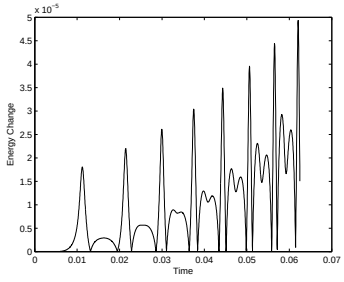
(e) Multipliers



(f) Multipliers cont.

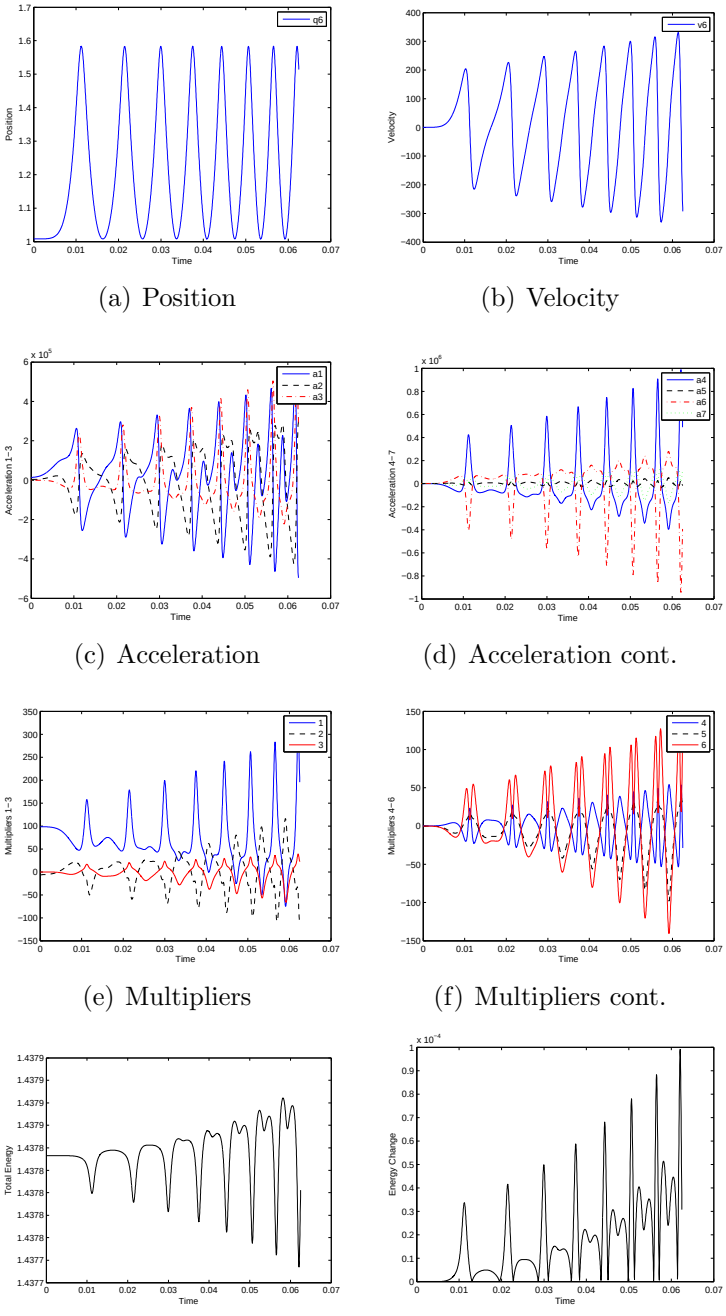


(g) Total Energy



(h) Energy Change

Figure A.35: 7 Bar Mechanism - Option 2 - Two Field Form



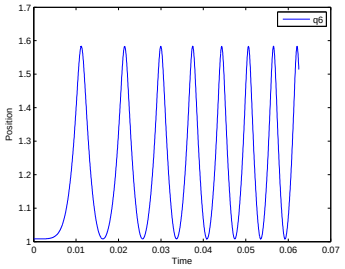
(a) Position (b) Velocity

(c) Acceleration (d) Acceleration cont.

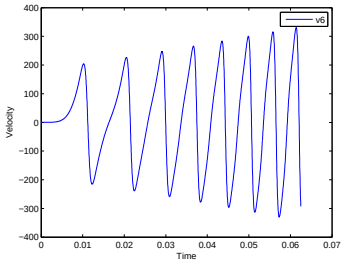
(e) Multipliers (f) Multipliers cont.

(g) Total Energy (h) Energy Change

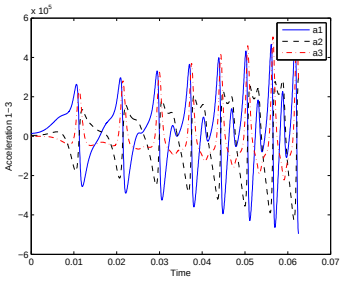
Figure A.36: 7 Bar Mechanism - Option 3 - Two Field Form



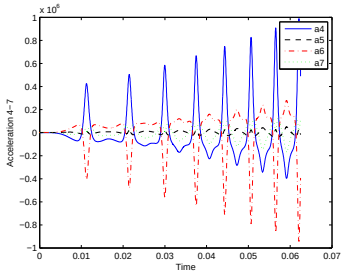
(a) Position



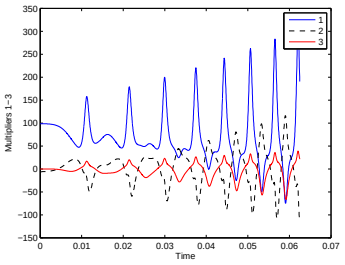
(b) Velocity



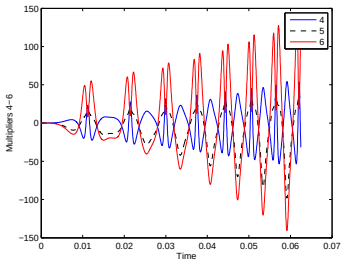
(c) Acceleration



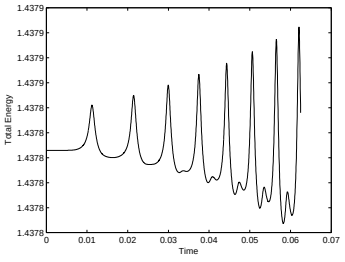
(d) Acceleration cont.



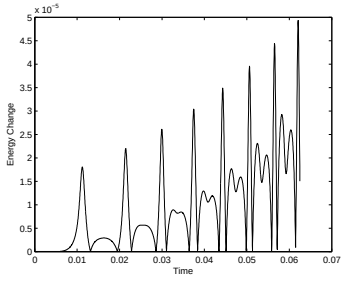
(e) Multipliers



(f) Multipliers cont.



(g) Total Energy



(h) Energy Change

Figure A.37: 7 Bar Mechanism - Option 4 - Two Field Form

Optimal Algorithm Constraint Satisfaction

To demonstrate the satisfaction of the position constraint and evaluation of the velocity and acceleration constraints the V0(1,1,0) algorithm was run for 50 seconds with a time step size of $\Delta t = 0.05$. The results can be seen below.

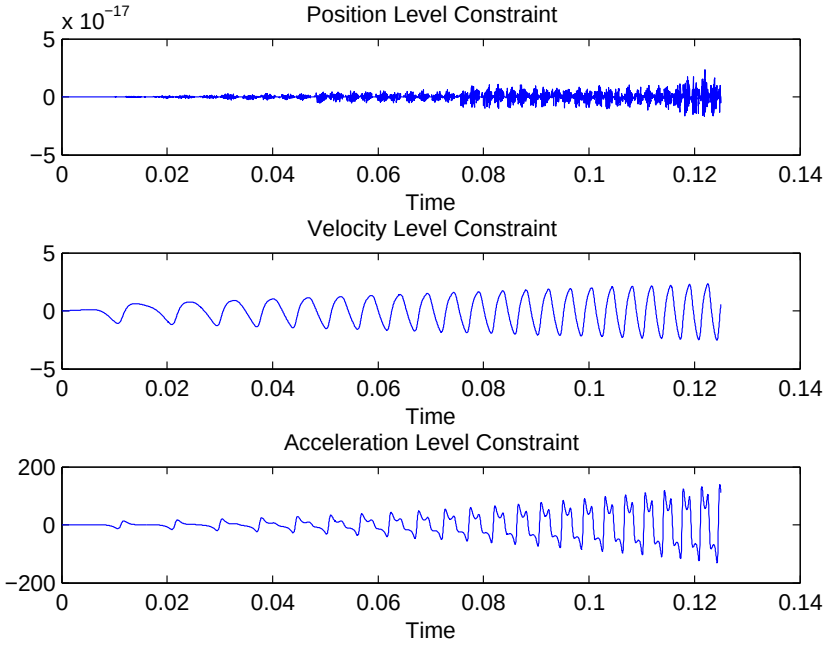


Figure A.38: 7 Bar Mechanism - Optimal Algorithm Constraints