MULTIBODY DYNAMICS PROBLEMS IN NATURAL COORDINATES

MULTIBODY DYNAMICS PROBLEMS IN NATURAL COORDINATES

THEORY, IMPLEMENTATION AND SIMULATION

BY

BEHRANG DERAKHSHAN, B.Sc.

A THESIS

SUBMITTED TO THE SCHOOL OF COMPUTATIONAL SCIENCE AND ENGINEERING

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

Master of Science (2022)	McMaster University
(computational science and engineering)	Hamilton, Ontario, Canada

TITLE:	Multibody Dynamics Problems in Natural Coordinates Theory, Implementation and Simulation
AUTHOR:	Behrang Derakhshan B.Sc. (Mechanical Engineering), Iran University of Science & Technology, Tehran, Iran
SUPERVISOR:	Dr N. Nedialkov
NUMBER OF PAGES:	xiii, 104

Abstract

We present a framework for modeling multibody systems based on the method of natural coordinates and Lagrange's equation of the first kind, resulting in a system of Differential-Algebraic Equations (DAEs). The C++ package DAETS (DAEs by Taylor Series), a robust high-index DAE solver, is utilized to solve the models. The simulation process is straightforward, with no need to derive equations of motion directly. Instead, the user supplies a Lagrangian, kinematic constraints, and if applicable, a dissipation function and external forces. A corresponding system of DAEs is formed by computing the required derivatives via automatic differentiation. DAETS primarily uses Cartesian coordinates as variables, eliminating angles and the associated trigonometric functions, which results in simplified models. Furthermore, DAETS provides direct access to the position/velocity data of any desired points or vectors as output, facilitating post-processing tasks, such as visualization. The main focus of this thesis is on establishing the viability of our framework through case studies. We simulate seven multibody systems and compare our results with those of reference models developed in the Simulink environment of MATLAB. A detailed account of the modeling process is given for each system, demonstrating the ease and intuitiveness of our approach. We also provide, from both DAETS and Simulink, the time history plots of several position coordinates to allow for direct comparison. Finally, we compute two types of errors over time. Our findings show that the results of DAETS match those of the reference models under different error tolerances for the studied systems, indicating that our framework is capable of simulating a wide variety of mechanisms with a superb degree of accuracy.

This thesis is dedicated with love to my parents and sister, whom I dearly miss, and also to the group of extraordinary people I am lucky enough to call friends: Shahin, Majid, Alireza, Sepehr, Shayan, Taha, Pouya, Sadra, Amir Mohammad, and last but not least, Hossein.

Contents

List of Figures

List of Tables

Notation, Definitions, and Abbreviations

Notation

a	scalar
$\mathbf{a}$	vector
A	position vector
$\mathbf{A}$	matrix

Abbreviations

w.r.t.	With Respect To
DoF	Degrees of Freedom
ODE	Ordinary Differential Equation
DAE	Differential-Algebraic Equation
IC	Initial Condition
IV	Initial Value
IVP	Initial Value Problem
PV	Point or Vector
BPV	Basic Point or Vector

2D	Two-dimensions or Two-dimensional
3D	Three-dimensions or Three-dimensional
WF	World Frame
LF	Local Frame
CoM	Center of Mass
MoI	Moment of Inertia
TC	Taylor Coefficient

Chapter 1

Introduction

Multibody systems pervade many aspects of engineering and science through their ubiquitous presence in mechanical systems, from something as mundane as an adjustable chair, or an instrument operating at the bleeding edge of technology and science, such as the fabled mirrors of the James Webb telescope [20]. Commensurate with this, to meet the ever-growing demand in both the academic and industrial scenes for improved methods to study multibody systems, this thesis focuses on a comprehensive framework for such studies.

At the core of our framework lies DAETS[1] [29, 30], a computational package implemented in C++ that makes available a set of tools for implementing and simulating multibody systems. We use Lagrange's equations in conjunction with a natural coordinates method to develop mathematical models for the purpose of tracking positions and velocities of rigid bodies. Unlike the Newtonian approach, the user does not need to derive the equations of motion directly. Instead, our system takes as input a Lagrangian, kinematic constraints, and potentially Rayleigh's dissipation function, as well as any external forces/torques if applicable, and then forms a system of differential-algebraic equations (DAEs) automatically.

The primary objective of this thesis is to demonstrate the viability of our framework. We perform case studies on seven mechanisms of varied complexities. First we create natural coordinates models, demonstrating the intuitive process our framework offers. Next we simulate our models for a specific

[1]Differential-Algebraic Equations by Taylor Series

set of operating conditions and present our results. Finally, we use multibody models developed in Simulink as a reference to first validate our model and then verify our results. This is achieved by computing and comparing two types of errors between DAETS' solution and that of Simulink, proving the ability of DAETS at accurately predicting the motion path of a mechanism.

The thesis is structured as follows. Chapter 1 provides an introduction to multibody systems and methods for their modeling and simulation. A review of past studies focusing on modeling approaches is presented. In Chapter 2 we detail our natural coordinates technique. Chapter 3 provides an overview of all the constituent parts of the framework and the pipeline that links them to create a cohesive whole. Chapter 4 introduces a general structure for the study of multibody systems using our framework. Based on this structure, we develop natural coordinates models for and simulate a number of planar and spatial systems in Chapters 5 and 6, respectively, validating our models and then verifying our results using reference solutions obtained from Simulink. Chapter 7 provides closing remarks, as well as recommendations for future works, particularly with regard to the development of DAETS.

1.1 Multibody Systems

Multibody is a term that encompasses a wide range of mechanical apparatus, such as robots, automobile steering and suspension linkages, and satellites [15]. In general, this term is used, along with *mechanism* and *linkage*, to refer to a collection of rigid and flexible bodies, force transmission components (springs, dampers, etc.), and various types of joints used to connect them, allowing for relative motion between the bodies.

Mechanisms fulfill a myriad of purposes, including: transmitting a force or torque from one point to another, achieving a desired motion trajectory at a specific point on a mechanism, and generating mechanical advantage. Due to their flexibility in performing various jobs, multibody systems are implemented in many fields, such as: the automotive industry, aerospace, healthcare, robotics, and manufacturing. This wide range of applications has resulted in extensive demand for multibody systems, creating a need for tools that facilitate their rapid design and analysis. Traditional methods of study, such as *graphical*, *analytical* and *experimental*, are of limited practical use when applied to complicated, real-world problems [15]. For instance, graphical methods offer a good understanding of linkage kinematics, but are also time-consuming and suffer from low accuracy [15]. Analytical methods, on the other hand, are highly efficient computationally, but tend to become overly complicated for three-dimensional (3D) applications [15]. Finally, carrying out experimental analysis (*i.e.*, physically testing a system in laboratory) is often not practical, since constructing a model that reproduces the characteristics and operating conditions of the original mechanism is usually difficult and costly.

In light of the mentioned limitations, the *computational* approach has emerged as a powerful alternative to the other methods. This technique involves using a computer to build a mathematical model of the system which is examined virtually, allowing for rapid experimentation under different scenarios to predict system behavior with great detail and little cost in terms of expenses and time. The following section deals with the computational approach in detail.

1.2 Computational Analysis of Multibody Systems

In the context of multibody systems, computational analysis can be split into two parts, namely *modeling* and *numerical simulation*. Modeling is the process of creating a mathematical model in the form of a system of purely differential (ODE) or differential-algebraic (DAE) governing equations that describes the kinematic and dynamic behavior of a mechanism and its elements. Such a system is comprised of the equations of motion plus any required complimentary relations that, together with a set of initial conditions (IC), form an initial value problem (IVP). Numerical simulation, on the other hand, is concerned with solving this problem by integrating the system of equations using numerical techniques, yielding the time history of mechanism's configuration.

There is no shortage of techniques for modeling a multibody system and solving the resulting equations. As such, while this section lays out a general outline for several methods used in current commercial and open-source multibody analysis programs, it only goes into detail for those approaches that are utilized in the subsequent chapters of the thesis.

1.2.1 The Mathematical Model

The behavior of a multibody system can be modeled mathematically using the following form of the Lagrange's equations of motion [29]

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_j} - \frac{\partial \mathcal{L}}{\partial q_j} + \sum_{i=1}^{n_c} \lambda_i \frac{\partial C_i}{\partial q_j} = Q_j(t), \quad j = 1, 2, \ldots, n_q, \tag{1.2.1}$$

$$C_i = 0, \quad i = 1, 2, \ldots, n_c, \tag{1.2.2}$$

where $\mathcal{L}$ signifies the *Lagrangian*, C_i are the set of *kinematic constraints*, Q_j denotes *generalized forces*, $\mathbf{q} = (q_1, q_2, \ldots, q_{n_q})$ with q_i being the the generalized coordinates, λ_i are Lagrange multipliers, and t denotes time. A brief description for these quantities is provided below.

Generalized coordinates vector. Equations (1.2.1) and (1.2.2) form a second-order DAE system in $\mathbf{q}$. Here $\mathbf{q} \in M$, the mechanism configuration space. For a multibody system operating in a Euclidean physical d-space, M is a subset of $\mathbb{R}^{dn_q}$. Considering a mechanism with N constituent elements, the position vector of every element in an inertial frame, $\mathbf{r}_k$, can be expressed [16] as a transformation from the configuration space to the physical space

$$\mathbf{r}_k = \mathbf{r}_k(q_1, q_2, \ldots, q_{n_q}, t), \quad k = 1, 2, \ldots, N.$$

An important aspect of the Lagrange's formulation is the non-uniqueness of the generalized coordinates vector. That is, the same multibody system can be modeled using different $\mathbf{q}$ vectors [15, 46]. Even though the solution to Lagrange's equation describes one unique motion in the physical space for a given mechanism and initial conditions, any suitable configuration space can be chosen according to one's needs. It also follows that different $\mathcal{L}, Q_j$ and C_i can be obtained for the same system. More details on choosing a set of coordinates is given in Section 1.3.

Lagrangian. This term is defined by

$$\mathcal{L} = T - V,$$

where T and V are the kinetic and potential energies of the system, respectively. Note that V contains the potential of conservative forces in the system, such as gravity and spring elastic forces, as well as the potential due to constant torques, whereas non-conservative forces and torques are accounted for in Q_j. Given the above, $\mathcal{L}$ will be a function of $\mathbf{q}, \dot{\mathbf{q}},$ and t.

Generalized forces. Q_j represents non-conservative forces and torques—*i.e.*, those not arising from a scalar potential function in the form of $V(\mathbf{q})$—such as dissipative forces. An important case is velocity-proportional frictional forces that occur frequently in multibody systems, usually stemming from energy-retarding elements such as dampers. Denoting the force vector by $\mathbf{F}$ we can write

$$\mathbf{F} = \big(-k_x v_x, -k_y v_y, -k_z v_z\big),$$

where k_x, k_y and k_z are the coefficients of friction in Cartesian directions, and v_x, v_y and v_z denote Cartesian velocity components. Such a force can be expressed using the Rayleigh dissipation function [16] as

$$\mathcal{R} = \frac{1}{2}\big(k_x v_x^2 + k_y v_y^2 + k_z v_z^2\big). \tag{1.2.3}$$

It is easy to see that

$$\mathbf{F} = -\nabla \mathcal{R}.$$

Rayleigh's dissipation function can be viewed as analogous to a potential field for velocity-dependent dissipative forces, and it can be incorporated into equation (1.2.1) using [16]

$$Q_j = -\frac{\partial \mathcal{R}}{\partial \dot{q}_j}.$$

Kinematic constraints. The components of a multibody system are restricted in their motion by physical constraints imposed by the intrinsic properties of the elements, such as rigidity conditions, or the presence of kinematic joints. These restrictions manifest in the model as a set of equations given by (1.2.2), rendering the coordinates dependent on each other.

If all constraints can be written explicitly as a function of the generalized coordinates and time—*i.e.*, $C_i = C_i(\mathbf{q}, t)$—the system is called *holonomic*, and the constraints are entirely algebraic. If derivatives of the generalized coordinates appear in the constraints, the system will be *nonholonomic*. In holonomic systems it is possible to find a set of generalized coordinates that contains the constraints implicitly, eliminating their equations entirely and turning it into an ODE system [16]; however, our model does not use this approach.

1.3 Approaches for Choosing a Set of Coordinates

An essential step in the modeling process of a multibody system is choosing a set of coordinates capable of uniquely determining the mechanism configuration. Finding such a set of coordinates has been a topic of interest [34] among multibody systems researchers, as it has a significant impact on the size and structure of the resulting model and, consequently, the computational performance of the solution process.

1.3.1 Independent Coordinates

One approach is to choose a set of *independent coordinates* [15] with as many elements as the number of system *degrees of freedom* (DoF)—the minimum number of independent scalar parameters required to fully determine the mechanism configuration. This leads to a system of second-order

ODEs. An advantage of this approach is that the obtained systems use a minimal set of coordinates, rendering them inexpensive to solve. However, care must be taken in selecting the individual coordinates so that they implicitly hold the constraints, otherwise the set of independent coordinates will not be capable of uniquely defining the system configuration, meaning that the mechanism can exist in more than a single state even after coordinate values have been determined [15].

1.3.2 Dependent Coordinates

The second approach is selecting a set of *dependent coordinates* [15] containing a greater number of elements than n_{DoF}—the number of DoF. A set of n_c constraint equations will then need to be added to the system. These constraints introduce dependencies between the coordinates, reducing the number of independent variables in the resulting DAE system. The three major types of dependent coordinates are summarized below.

Relative Coordinates

In this technique [15, 34, 35, 43], a mechanism is viewed as a series of bodies connected by joints, forming a *kinematic chain*. The position of each element is defined relative to the previous element in the chain via coordinates that correspond to the relative DoF due to the joint connecting the bodies. For example, if we consider the case of a planar mechanism, the relative position of the two bodies connected by a revolute joint is defined by an angle, since revolute joints allow for a single relative rotational DoF.

An advantage of this method is that the system of equations generated has the minimum number of coordinates compared to other dependent coordinates methods. In the case of open kinematic chains—mechanisms in which no closed loops can be identified—the coordinate count even matches the number of DoF, eliminating constraint equations entirely [15]. However, there is a trade-off in that lower system size usually leads to an increase in nonlinearity, and the two can balance each other out [46] in terms of computation speed and memory requirements. Also, reducing system size does not necessarily lead to increased numerical efficiency. Important drawbacks of relative coordinates include: complex and hard-to-read formulations, and the density of the matrices present in the equations of motion, making them computationally expensive to evaluate [15].

Reference Point Coordinates

Reference point coordinates [15,46] attempt to alleviate some of the issues of the relative coordinates method by directly keeping track of the position and orientation of every element in the system in relation to a fixed (inertial) reference frame. Position is defined using the Cartesian coordinates of a reference point on the element, while a set of angles (*e.g.*, Euler angles [36]) determine the angular orientation of a *local frame* rigidly attached to the body w.r.t. the inertial frame. The nature of this method enables one to derive formulations that are easier to read and understand. Furthermore, the resulting systems of equation tend to be much sparser than those based on relative coordinates; hence, given that appropriate numerical treatment is applied, computational costs for solving the system can be lower [46].

Natural Coordinates

The method of natural coordinates [10, 15] is based on representing the elements of the system as a distribution of points or vectors (PVs). The coordinates consist solely of the Cartesian coordinates of these PVs or a linear combination of them, eliminating the angular variables used in the previous methods. Moreover, the PVs can be shared between bodies at joints instead of explicitly defining a constraint equation, reducing system size. As a result, the number of parameters in natural coordinates tends to fall in between that of relative and reference point coordinates [15]. Some of the other characteristics of natural coordinates include:

- Using Cartesian coordinates results in more readable equations that are easier to understand.
- The formulation lends itself to the programmatic implementation of the equations.
- The coordinates are defined directly in the inertial reference frame, giving the global positions at all times, making the results easy to visualize.
- If the system is modeled using two points and two vectors to represent each body, the inertia mass matrix will be constant.

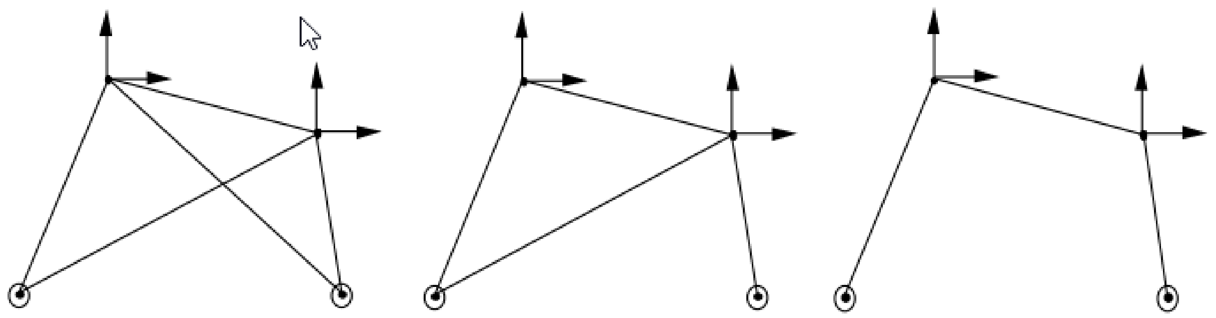


Figure 1.1: Three possible configurations for a planar truss; from [10, p. 17].

As an approach for modeling multibody systems, natural coordinates has gone through a history of evolution since its inception. In fact, this method originally started out as an adapted version of the *displacement method* used for matrix analysis in 2D structures [10]. The planar truss is taken as a sample structure, for which Figure 1.1 shows different configurations. The dots represent revolute joints, while the lines are bars. There are two moving nodes at the top, the position of which can be described using four Cartesian coordinates. The system has four equilibrium equations, which can be solved to obtain the (unknown) reaction forces in the bars. The difference between the three conditions in Figure 1.1 stems from the disparity in the number of bars. The left-hand side configuration is overdetermined in the sense that it has five unknown reaction forces; the middle truss with four bars is a determined system since the number of unknown forces is the same as the number of equations; finally, the right-hand truss is an underdetermined system, having only three reaction forces, and is thus an unstable structure [10]. It is easy to see that it is actually a four-bar mechanism.

The displacement method forms the equilibrium relations by equating external forces with the elastic reaction forces of the deformable bars [10]:

$$\mathbf{K}\mathbf{x} = \mathbf{f}, \quad \mathbf{K} = \sum_{i=1}^{4} \mathbf{K}_i^e,$$

where $\mathbf{K}$ is the structure stiffness matrix, $\mathbf{x}$ is the displacements vector, and $\mathbf{f}$ denotes the external forces. In the case of an unstable structure, $\mathbf{K}$ will be singular due to the fact that displacement can occur even in the absence of external forces or bar deformations. These displacements can be categorized as rigid body motions because they produce neither elastic energy nor deformation. The null space of $\mathbf{K}$ contains all such displacements [10].

Four decades ago, researchers began building upon the concept of using the stiffness matrix and its null space, developing novel methods for performing kinematic analysis in Cartesian coordinates. The pioneering papers in describing these new formulations were authored by Serna and Avileś [11, 41]. Around the same time, the first article to solely use Cartesian coordinates for kinematic analysis in spatial multibodies was published by de Jalón *et al.* [12]. The authors considered three types of geometric constraints: constant distance of two points, constant area of a triangle formed by three points, and the constant volume of a tetrahedron made of four points, assuming all points lie on a rigid body. These basic relations allowed for rigidity and many lower pair joint constraints to be readily formulated. Revolute and spherical joints were handled by sharing one and two points between bodies, respectively.

The next step in the evolution of natural coordinates was the addition of the Cartesian components of unit vectors to the list of parameters by Vilallonga *et al.* [45]. One interesting aspects of their work was transitioning from the least squares formulation of the constraints to one based on vector products. The first natural coordinates formulation for 3D multibody dynamics was published in 1986 by de Jalón *et al.* [13]. They described the inertia forces of a rigid body using two points, two non-coplanar unit vectors, and a local reference frame attached to the body. This approach was also utilized to obtain the global position of points of interest. Their method enabled the definition of velocity-independent inertia forces as well as a constant mass matrix. The matrix components were a function of mass, the position of center of gravity in the local frame, and the inertia tensor of the body. An important feature of this method is that the constant mass matrix can be taken advantage of for numerical efficiency [4].

The natural coordinates method has seen many developments since the work of de Jalón, resulting in smaller equation sets, more efficient formulations leading to a decrease in computational costs, and so forth. Some of these methods and their implementation are presented below.

Von Schwerin [46] makes use of two points and two unit vectors in 3D, a total of 12 Cartesian coordinates, per rigid body, in combination with the descriptor form of the equations of motion—an index-3 [6] DAE given by

$$\begin{bmatrix} \mathbf{M} & \boldsymbol{\Phi}^{\mathbf{T}} \\ \boldsymbol{\Phi} & 0 \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{Q} \\ \mathbf{c} \end{bmatrix}, \quad \mathbf{c} = -\dot{\boldsymbol{\Phi}}_{\mathbf{q}}\dot{\mathbf{q}},$$

where $\mathbf{M}$ is constant, $\boldsymbol{\Phi}$ is the constraints Jacobian, $\boldsymbol{\lambda}$ is the vector of Lagrange multipliers, and $\mathbf{Q}$ denotes the generalized forces vector. The null space method (NSM) and range space method (RSM) are the two primary techniques for solving the above quasilinear (*i.e.*, linear in $\ddot{\mathbf{q}}$) system. According to von Schwerin, the NSM can be up to 3.5 times faster than RSM for heavily constrained systems—where the DoF is lower than half the number of $\mathbf{q}$ components. Furthermore, the NSM offers another benefit in dealing with accelerations based on independent coordinates, enabling the use of efficient explicit integrators such as Adams-Bashforth-Moulton in non-stiff systems. As a whole, it is claimed that a complexity of $O(n)$ can be achieved using the descriptor form, with n denoting system size.

Von Schwerin also examined the efficiency of two numerical integrators used to solve the equations of motion for a spatial five-link suspension system, the very same mechanism we also simulate as our final problem in chapter 3, in both natural and relative coordinates. The first one is the ADAMS [40] forward dynamics solver MBSABM, the results of which do not show a significant difference between the natural and relative formulations in terms of required CPU time over a wide range of accurate digits count. This is taken as evidence that the performance gain from lower system dimensions (relative coordinates) is roughly equal to the gain from the reduction in system nonlinearity (natural coordinates), leading to almost identical numerical efficiencies. The second solver is MBSIDY, an inverse dynamics integrator also contained in ADAMS. Von Schwerin reports that using this solver, as opposed to the previous case, the natural coordinates model has a 40% lower CPU time compared to the relative coordinates model, because the integrator can take advantage of the simpler formulation. More details can be found in [46, section 1.7].

Considering the above, Von Schwerin arrives at the conclusion that it is prudent not to measure separately the effect of the choice of coordinates and numerical treatment on computational performance, but to view them as a whole; because, for instance, an integrator that results in a significant amount of increase in efficiency for one model may even be detrimental to performance for another. He also states that when attempting to reduce the number of model variables in pursuit of performance gain, one must take care to preserve the structure of equation that a numerical solver might exploit.

Cardenal *et al.* [7] developed an efficient natural coordinates model with an augmented Lagrangian formulation [19] that uses the descriptor form of the equations of motion with two formulations. Their solver utilizes a variable time step trapezoidal rule for integration, taking advantage of the better performance of the index-3 formulation for larger time steps, and the superior stability properties of the index-1 formulation for smaller steps. This approach results in a natural coordinates model with a constant mass matrix and quadratic constraint relations, which is usable for both stiff and non-stiff problems.

Cuadrado *et al.* [9] examined the dynamics of a 4×4 military vehicle performing complex maneuvers that required a large number of coordinates with a focus on finding suitable methods for real-time analysis of complex multibody systems. They compared the performance of their model with two others: a fully-recursive, relative coordinates model, and a classic index-3 DAE formulation based on the Lagrange method with no augmentation. In addition to the impact of different mathematical formulations, the effect of numerical techniques, such as taking advantage of the sparsity of the system of equations, as well as other factors, such as parallel computing, is inspected. Interestingly, it is found that the fully-recursive model does not benefit from parallelization at all, whereas the augmented Lagrangian profits the most, having its CPU time reduced by about 30% using four processing units. The paper proposes that the augmented formulation is a general-purpose model suitable for a wide variety of applications; in contrast, the relative coordinates model is presented as more suited for open-chain mechanisms.

In the works of Kraus [21, 22, 23], each rigid body is tracked using one point and three unit vectors, enabling the definition of a constant mass matrix. This method does not take advantage of point sharing between bodies—which is a major benefit of natural coordinates as it leads to eliminating constraint equations from the system. By considering system topology and using a partitioning method for handling rank-deficient constraint Jacobians, Kraus arrives at an algorithm with linear complexity in the number of rigid bodies. This approach is then used for parameter estimation on a human body model with 82 kinematic DoF.

Simulating a motorcycle is the main focus of the work of Cossalter & Lot [8]. They use a natural coordinates model with 11 DoF to simulate the front and rear chassis, suspension and steering systems, as well as the tires. A local frame is attached to each body, adding the Cartesian coordinates of its origin plus three direction cosines as variables to the system. After forming an index-3 DAE system using Lagrange's formula, it is transformed into an index-1 DAE through replacing the algebraic constraints by differential equations via Baumgarte stabilization method [3]. The paper presents results that match very well with those of an experimental model. The computational efficiency of the natural coordinates method is also compared with commercial software, namely DADS [44] and Visual Nastran. The authors show that the natural coordinates method is much faster, to the point that it is suitable for real-time simulation applications.

In contrast to all the aforementioned research, the work of Pryce & Nedialkov adopts a different natural coordinates method that uses not four, but three PVs per body, along with PV sharing between them. This enables an intuitive definition of body-fixed local reference frames, making complex mechanisms easier to model and visualize. It also leads to a smaller set of coordinates. After forming the Lagrangian term, automatic differentiation [39] (instead of symbolic manipulation) is utilized to convert it to an index-3 DAE with a non-constant inertia mass matrix. A high-performance C++ code called DAE by Taylor series (DAETS) [30, 31] developed by the authors is used to integrate the equations, effectively offsetting the loss of constancy in $\mathbf{M}$ and alleviating concerns of dealing with a high-index DAE. The paper also presents an original tool called *MechSpec* for defining multibody systems using YAML files, which can be fed to the *Mechanism Facility* in order to automatically form a natural coordinates model of the mechanism that is solvable by DAETS.

Chapter 2

The Modeling Framework

We aim to create a natural coordinates framework for modeling a multibody system with the goal of simulating the dynamical behavior of system elements for a given set of initial and operating conditions. This requires obtaining the time history of the position and orientation of all elements to determine the system configuration at time t. It is important to note that only rigid bodies are considered in this study. The theory and formulae presented in this chapter are either taken from or based on the work of Pryce & Nedialkov [38], unless stated otherwise.

This chapter is structured as follows. Section 2.1 lays out the basic assumptions and concepts used in developing the model. Sections 2.2 and 2.3 develop the relationships used for tracking the positions and velocities of mechanism elements. Section 2.4 discusses the Lagrangian term, while Section 2.5 talks about constraint equations and how our model implements them. Finally, Section 2.6 provides detail on how dissipative forces are incorporated into the model.

2.1 Basic Conventions

The process of building our natural coordinates (NC) model begins by defining the world in which the mechanism operates. It is a Euclidean d-space, with d being 2 or 3 in practice. We choose an arbitrary point in the world as the origin for an orthonormal basis inertial reference frame E^d fixed in space—the *world frame* (WF). The term *global* is used to refer to variables measured in the WF.

The motion of every point on a rigid body $\mathcal{R}$ in the WF is defined by

$$\mathcal{R}_t = \mathsf{O}_t + \mathbf{Q}_t\mathcal{R}, \tag{2.1.1}$$

where O_t is the global position of an arbitrary point fixed on the body, and $\mathbf{Q}_t$ is the frame's rotation matrix, detailed in Section 2.2. Quantities bearing the subscript $_t$ are a function of time—*i.e.*, they are measured w.r.t. the WF. Hence, in general, they vary over time. On the contrary, $\mathcal{R}$ (without the subscript) is measured in relation to a body-fixed copy of the WF with its origin placed at O, called the *local frame* (LF), and is thus constant due to rigidity requirements. Each body has its own local frame. From (2.1.1) it follows that the motion of any point X or vector $\mathbf{u}$ fixed on $\mathcal{R}$ is given by

$$\mathsf{X}_t = \mathsf{O}_t + \mathbf{Q}_t\mathsf{X}, \quad \mathbf{u}_t = \mathbf{Q}_t\mathbf{u}, \tag{2.1.2}$$

where X and $\mathbf{u}$ are defined in the local frame. Equation 2.1.2 converts the coordinates of a PV from the local frame to the WF.

Next we introduce the generalized coordinates vector $\mathbf{q}$ and its time rate of change $\dot{\mathbf{q}}$. The members of $\mathbf{q}$ are the global Cartesian coordinates of PVs residing on rigid bodies. For the rest of this thesis, capital X, Y and Z subscripts only denote Cartesian components as measured w.r.t. the WF.

2.2 Tracking Bodies in Spatial Systems

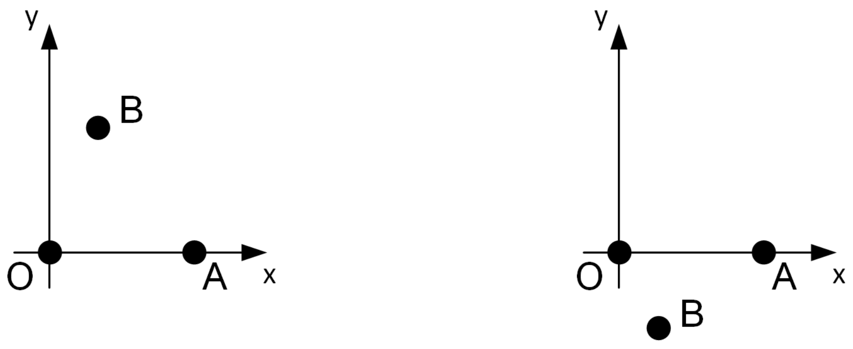
Position tracking. The body-fixed reference frames defined in the previous section are used to find the position, orientation and velocity of a rigid body in the WF, referred to as *tracking* for brevity. For the 3D case, creating a local frame needs an ordered list of three non-collinear *basic* PVs (BPVs) residing on the body, which we will denote by O, A, and B. The frame is defined according to these rules:

- O must be a point and serves as the origin.
- A can be either a point or vector and is on or parallel to the local x-axis.
- B can be either a point or vector and is on the local xy plane.

We can arrange the local coordinates of the BPVs in a square matrix $\mathbf{P}$ as column vectors:

$$\mathbf{P} = [\mathsf{O}, \mathsf{A}, \mathsf{B}] = \begin{bmatrix} 0 & r_{11} & r_{12} \\ 0 & 0 & r_{22} \\ 0 & 0 & 0 \end{bmatrix} = \left[\begin{array}{c|c} \begin{matrix} 0 \\ 0 \end{matrix} & \mathbf{R} \\ \hline 0 & 0 \quad 0 \end{array}\right]. \tag{2.2.1}$$

$\mathsf{O}, \mathsf{A},$ and B must be chosen in a way that renders matrix $\mathbf{R}$ nonsingular. The positive direction of the x and y axes can be adjusted by changing the sign of r_{ii}. Examples are provided in Figure 2.2.



(a) $r_{11}, r_{22} > 0$ (b) $r_{11} > 0$, $r_{22} < 0$

Figure 2.2: Local reference frame orientation examples.

To compute the rotation matrix, a systematic way of obtaining the global coordinates of the frame's basis vectors $\widehat{\mathbf{x}}_t, \widehat{\mathbf{y}}_t,$ and $\widehat{\mathbf{z}}_t$ is needed. In the local frame, we can write $\widehat{\mathbf{x}} = \mathsf{A} - \mathsf{O}$, given A is a point. The same is true for $\widehat{\mathbf{y}}$ and B. These subtractions can be written in matrix form as $\mathbf{PS}$, with

$$\mathbf{S} = \begin{bmatrix} -\alpha & -\beta \\ 1 & 0 \\ 0 & 1 \end{bmatrix}; \qquad \alpha = \begin{cases} 1 \text{ if } \mathsf{A} \text{ is a point} \\ 0 \text{ if } \mathsf{A} \text{ is a vector} \end{cases}$$
$$\text{similar treatment for } \beta \text{ and } \mathsf{B}.$$

Now we can obtain the basis vectors in the local frame by defining $\mathbf{U} = \mathbf{S}\mathbf{R}^{-\mathbf{1}}$, yielding:

$$[\widehat{\mathbf{x}}, \widehat{\mathbf{y}}] = \mathbf{P}\mathbf{U} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}. \tag{2.2.2}$$

The time-dependent counterpart of matrix $\mathbf{P}$ can be written as

$$\mathbf{P}_t = [\mathsf{O}_t, \mathsf{A}_t, \mathsf{B}_t].$$

From (2.1.2) it can be concluded that equation (2.2.2) must hold using global coordinates as well [38], leading to

$$[\widehat{\mathbf{x}}_t, \widehat{\mathbf{y}}_t] = \mathbf{P}_t\mathbf{U} = [\mathsf{O}_\mathbf{t}, \mathsf{A}_\mathbf{t}, \mathsf{B}_\mathbf{t}]\,\mathbf{U}. \tag{2.2.3}$$

Now, $\mathbf{Q}_t$ can be formed as a 3×3 matrix with $\widehat{\mathbf{x}}_t$ and $\widehat{\mathbf{y}}_t$ as the first two columns. The final basis vector serves as the third column, and it can be computed using a cross product, resulting in a positively-oriented frame:

$$\widehat{\mathbf{z}}_t = \widehat{\mathbf{x}}_t \times \widehat{\mathbf{y}}_t. \tag{2.2.4}$$

Finally, the rotation matrix is obtained as

$$\mathbf{Q}_t = [\widehat{\mathbf{x}}_t, \widehat{\mathbf{y}}_t, \widehat{\mathbf{z}}_t].$$

The Cartesian components of the PVs used to create local frames are either in the generalized coordinates vector or can be computed from it. By defining a body-fixed frame and computing the corresponding rotation matrix, the position of every PV on the rigid body can be calculated from (2.1.2). Obtaining the linear and angular velocities, however, requires further derivations outlined below.

Velocity tracking. In a similar manner, we can compute the linear and angular velocities for every point on a rigid body via a local frame. Using the same PVs as in (2.2.1) and the frame they establish, the time rate of change of the basis vectors in the WF is obtained by differentiating (2.2.3) and (2.2.4) as

$$\left[\dot{\widehat{\mathbf{x}}}_t, \dot{\widehat{\mathbf{y}}}_t\right] = \left[\dot{\mathsf{O}}_t, \dot{\mathsf{A}}_t, \dot{\mathsf{B}}_t\right] \mathbf{U},$$
$$\dot{\widehat{\mathbf{z}}}_t = \dot{\widehat{\mathbf{x}}}_t \times \widehat{\mathbf{y}}_t + \widehat{\mathbf{x}}_t \times \dot{\widehat{\mathbf{y}}}_t;$$

whereupon $\dot{\mathbf{Q}}_t$ can be formed as

$$\dot{\mathbf{Q}}_t = \left[\dot{\widehat{\mathbf{x}}}_t, \dot{\widehat{\mathbf{y}}}_t, \dot{\widehat{\mathbf{z}}}_t\right].$$

The angular velocity tensor $\mathbf{W}$ is related to the rotation matrix [14] by:

$$\mathbf{W} = \mathbf{Q}_t{}^T \dot{\mathbf{Q}}_t = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}, \tag{2.2.5}$$

where $\boldsymbol{\omega}_t = (\omega_x, \omega_y, \omega_z)$ is the body's angular velocity measured about the local frame axes. From (2.2.5) we have

$$\boldsymbol{\omega}_t = \begin{bmatrix} \widehat{\mathbf{z}}_t \cdot \dot{\widehat{\mathbf{y}}}_t \\ \widehat{\mathbf{x}}_t \cdot \dot{\widehat{\mathbf{z}}}_t \\ \widehat{\mathbf{y}}_t \cdot \dot{\widehat{\mathbf{x}}}_t \end{bmatrix}.$$

With the frame's angular velocity at hand, we move on to linear velocity, which for all PVs on $\mathcal{R}$ can be determined by simply differentiating (2.1.2):

$$\dot{\mathsf{X}}_t = \dot{\mathsf{O}}_t + \dot{\mathsf{Q}}_t\mathsf{X}, \quad \dot{\mathbf{u}}_t = \dot{\mathsf{Q}}_t\mathbf{u}. \tag{2.2.6}$$

2.3 Tracking Bodies in Planar Systems

Position tracking. The general procedure for creating body-fixed local frames in 2D is similar to the 3D case but much simpler, because it requires forming only two basis vectors, which necessitates designating two BPVs on a body. They are denoted by O and A; O must be a point and serves as the origin, while A is on the local x-axis. As a result, the matrix holding the local coordinates of these BPVs, namely $\mathbf{P}$, becomes

$$\mathbf{P} = [\mathsf{O}, \mathsf{A}] = \begin{bmatrix} 0 & r_{11} \\ 0 & 0 \end{bmatrix},$$

and $R = r_{11}$ is now a scalar. Matrix $\mathbf{S}$ becomes

$$\mathbf{S} = \begin{bmatrix} -\alpha \\ 0 \end{bmatrix}, \qquad \alpha = \begin{cases} 1 \text{ if } \mathsf{A} \text{ is a point} \\ 0 \text{ if } \mathsf{A} \text{ is a vector} \end{cases}.$$

With $\mathbf{U} = \mathbf{S}r_{11}^{-1}$, the global coordinates of the first basis vector (x-axis) is obtained from

$$\widehat{\mathbf{x}}_t = \mathbf{P}_t\mathbf{U} = \begin{bmatrix} u_t \\ v_t \end{bmatrix}.$$

The second basis must be perpendicular to the first, so it takes the form

$$\widehat{\mathbf{y}}_t = \begin{bmatrix} -v_t \\ u_t \end{bmatrix},$$

leading to the rotation matrix

$$\mathbf{Q}_t = [\widehat{\mathbf{x}}_t, \widehat{\mathbf{y}}_t]\,. \tag{2.3.1}$$

Velocity tracking. The velocities in 2D follow Equation (2.2.6) as before. The temporal derivative of the rotation matrix is also obtained in the same manner:

$$\dot{\mathbf{Q}}_t = [\dot{\widehat{\mathbf{x}}}_t, \dot{\widehat{\mathbf{y}}}_t].$$

The angular velocity tensor then becomes

$$\mathbf{W} = \mathbf{Q}_t{}^T\dot{\mathbf{Q}}_t = \begin{bmatrix} 0 & -\omega_t \\ \omega_t & 0 \end{bmatrix},$$

with $\omega_t = \widehat{\mathbf{y}} \cdot \dot{\widehat{\mathbf{x}}}$.

Hereafter we drop the $_t$ subscript for PVs, rotation matrices, and their derivatives that are measured w.r.t. WF, when this fact is clear from the context of the formula being presented.

2.4 The Lagrangian

The Lagrangian is defined by convention as the difference between the total kinetic and potential energies of the system. What follows is a definition of each energy term.

Kinetic energy. Kinetic energy can be split into two separate terms, namely translational and rotational kinetic energies. If the local position vector of the center of mass (CoM) of $\mathcal{R}$ is denoted by C, from (2.1.2) and (2.2.6) the global position and velocity of C are given by

$$\mathsf{C}_t = \mathsf{O}_t + \mathbf{Q}_t\mathsf{C} \quad \text{and}$$
$$\dot{\mathsf{C}}_t = \dot{\mathsf{O}}_t + \dot{\mathbf{Q}}_t\mathsf{C},$$

respectively. Now, the kinetic energy of $\mathcal{R}$ for spatial systems can be obtained as

$$T = \underbrace{\frac{1}{2}m\dot{\mathsf{C}}_t^2}_{\text{translational}} + \underbrace{\frac{1}{2}\boldsymbol{\omega}_t^T\mathbf{I}\boldsymbol{\omega}_t}_{\text{rotational}},$$

where m is the body's mass and $\mathbf{I}$ is the mass moment of inertia (MoI) matrix measured about a set of axes parallel to the local frame that pass through the CoM. In planar mechanisms, the rotational term changes to

$$T_{\text{rotational}} = \frac{1}{2}I\omega_t^2,$$

with I being the scalar MoI about the CoM.

Potential energy. We consider two types of potential energies stemming from gravity and spring elastic forces.

- **Gravity:** Assuming a gravitational acceleration vector $\mathbf{g} = \big(0, 0, -g\big)$ in the WF, the potential energy term for each rigid body due to gravity is given by

$$V_{\text{gravitational}} = mg\mathsf{C}_z,$$

where C_z is the z-component of the global position vector of $\mathcal{R}$'s CoM.

- **Spring elasticity:** The potential energy stored in a spring depends on the current length, which can be written as a function of the global position of the spring's endpoints. Assuming a spring connecting points A and B with stiffness k and natural length ℓ_0, the potential energy is given by

$$V_{\text{spring}} = \frac{1}{2}k\big(\big|\mathsf{A} - \mathsf{B}\big| - \ell_0\big)^2,$$

where the $\big|\cdot\big|$ operator gives the current length of the operand.

All the position variables used in the relations above are a function of $\mathbf{q}$, and so is the potential energy term as a whole.

2.5 Constraint Equations

Kinematic constraint equations in our natural coordinate model arise from rigidity constraints, as well as joints-imposed restrictions. We do not deal with nonholonomic constraints (equations that involve derivatives of general coordinates) in this work.

Rigidity conditions. Two points A and B residing on the same rigid body always have the same distance. This can be imposed using a quadratic equation that fixes the length of the vector connecting the points:

$$\big|\mathsf{A} - \mathsf{B}\big|^2 - \ell^2 = 0,$$

where ℓ is the distance between A and B.

Another property of rigid bodies is that the angle between two vectors lying on the body does not change. By choosing two vectors $\mathbf{u}$ and $\mathbf{v}$ fixed on the body that are perpendicular to one another, we can write the corresponding constraint as

$$\mathbf{u} \cdot \mathbf{v} = 0,$$

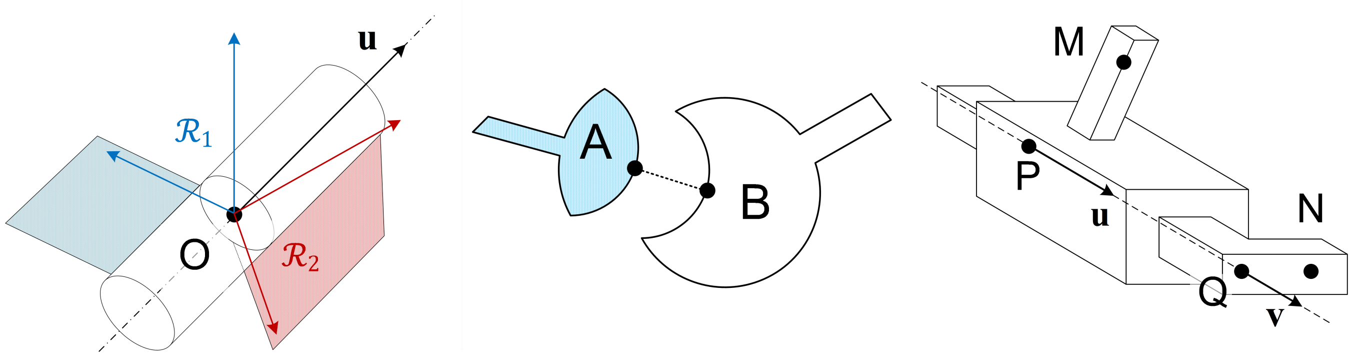
where $\cdot$ is the inner product operator.

Joint-imposed conditions. Kinematic joints connect two rigid bodies and, in the process, restrict their relative motion, which manifests itself as a reduction in model DoF. In a natural coordinates model, one can either explicitly reduce DoF by adding constraint equations, or remove appropriate components of $\mathbf{q}$ by way of sharing BPVs between local frames on different bodies, effectively imposing a constraint implicitly.

Joints can be categorized into *lower pair*, *higher pair*, and *compound* types. Lower pairs, such as revolute, prismatic and spherical joints, provide an area of contact between bodies and are usually more restrictive in terms of DoF compared to other joint types. Higher pairs create a point or line contact and include the rolling cylinder, cam joint, etc. Compound joints, as their name suggests, are a combination of one or more basic joints; for example, a universal joint is a compound pair consisting of two revolute joints with perpendicular rotation axes. The constraints imposed by the joints encountered in Chapters 2 and 3 are introduced below. An illustration of the joints in the 3D case is given in Figure 2.3, in which $\mathcal{R}_i$ marks the local frames of the bodies connected by joints.

- **Revolute**: A revolute joint connects two bodies at a contact point, denoted by O, restricting their relative motion to rotation about an axis $\mathbf{u}$ that passes through O. Here, our natural coordinates model avoids using constraint equations, and instead shares O and $\mathbf{u}$ as common PVs in the creation of each body's local frame to implement the joint, as shown in Figure 2.3a.

- **Spherical:** Also called a ball joint, this joint links two bodies at a common point, removing three translational DoF (two in the 2D case). We model a spherical joint simply by sharing a point between the bodies so that the only possible relative motion is rotation about this



(a) Revolute joint. (b) Spherical joint. (c) Prismatic joint.

Figure 2.3: Kinematic joint schematics—the 3D case.

point, removing the need for explicit use of constraints equations.

- **Prismatic** A prismatic or slider joint limits bodies to pure translation along a predefined path, providing one DoF. One way of applying a prismatic joint is assigning points P and M as well as vector $\mathbf{u}$ to one of the bodies, whilst designating points Q and N, plus vector $\mathbf{v}$ on the other [15]. Figure 2.3c illustrates the positioning of these PVs. For a sliding motion to take place, $\mathbf{u}$ and $\mathbf{v}$ must be parallel, enforced by a cross product, yielding two independent constraint relations:

$$\mathbf{u} \times \mathbf{v} = \mathbf{0}.$$

P and Q should also be aligned with the vectors, leading to

$$\left(\mathsf{P} - \mathsf{Q}\right) \times \mathbf{u} = \mathbf{0},$$

which gives two more independent equations. Finally, in order to eliminate relative rotation, an additional scalar product relation is needed to keep the angle between $\mathsf{M} - \mathsf{P}$ and $\mathsf{Q} - \mathsf{N}$ constant, which provides one equation:

$$(\mathsf{M} - \mathsf{P}) \cdot (\mathsf{Q} - \mathsf{N}) - \rho = 0,$$

where ρ is a constant computable from system initial state. From the total of five (two in the 2D case) scalar equations above, two can be implicitly implemented by sharing $\mathbf{u}$ between $\mathcal{R}_1$ and $\mathcal{R}_2$.

2.6 Rayleigh's Dissipation Function

We consider friction-like dissipative forces that are proportional to velocity via explicitly defining them using Rayleigh's dissipation function, defined in (1.2.3). Such forces are often generated by a linear or rotational damper. The case of a linear damper connecting points A and B is depicted in Figure 2.4a. The magnitude of the damping force [15] is obtained as

$$F_{f,\text{ linear}} = c(\dot{\mathsf{A}} - \dot{\mathsf{B}}) \cdot \mathbf{u},$$

where c is the linear damping coefficient and the corresponding Rayleigh function will have the form

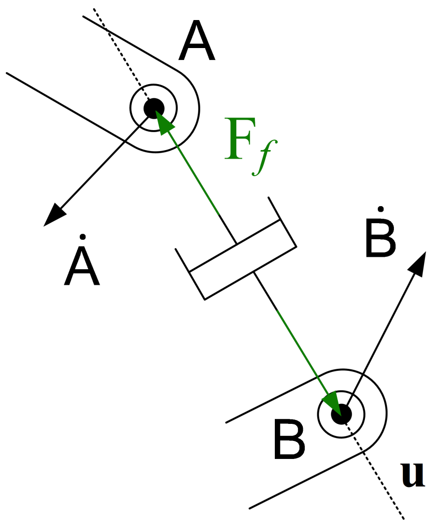
$$\mathcal{R} = \frac{1}{2}c\left((\dot{\mathsf{A}} - \dot{\mathsf{B}}) \cdot \mathbf{u}\right)^2,$$

where

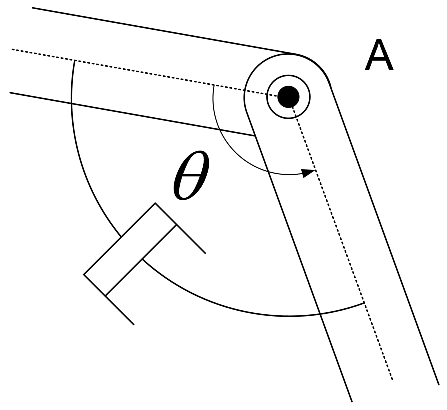
$$\mathbf{u} = \frac{\mathsf{A} - \mathsf{B}}{|\mathsf{A} - \mathsf{B}|}.$$

The dot product projects the velocity difference vector onto $\mathbf{u}$, the unit vector of $\mathsf{B}-\mathsf{A}$—known as the damper's line-of-sight [24]. Its inclusion in the formula is necessary because the force generated by a linear damper is proportional to velocity difference, and also acts, along the damper's line-of-sight. For the rotational damper shown in Figure 2.4b, the damper exerts a dissipative force with a magnitude [15] of

$$F_{f,\text{ rotational}} = c_{\text{r}}\dot{\theta}$$



(a) A linear damper.



(b) A rotational damper.

Figure 2.4: Schematic of damper types supported by DAETS.

on the rods, and the corresponding Rayleigh's function is given by

$$\mathcal{R} = \frac{1}{2} c_{\mathrm{r}} \dot{\theta}^2,$$

with c_{r} being the rotational damping coefficient. The $\dot{\theta}$ appearing in equations above does not need to be included in the $\dot{\mathbf{q}}$ vector. Instead, it can be computed using Cartesian coordinates from $\mathbf{q}$ and $\dot{\mathbf{q}}$.

Chapter 3

An Overview of Simulation Workflow

This chapter provides details on the systematic approach that our framework follows for the simulation of multibody systems. The workflow is divided into a number of blocks, depicted in Figure 3.5. Section 3.1 deals with mechanism specification and how the user encodes the multibody system of interest by supplying generalized coordinates, the Lagrangian, and so forth. Section 3.2 explains how the Lagrangian facility of DAETS works. Next, we present a concise discussion on the workings of the numerical engine of DAETS in Section 3.3. Finally, Section 3.4 talks about the features of the output facility, tasked with reporting information on the DAE system, as well as the numerical results.

3.1 Mechanism Specification

In accord with the natural coordinates approach detailed in Chapter 2, the modeling framework expects the mechanism to be specified as a group of BPVs distributed on system elements, the coordinates of which are to be provided as a set of generalized coordinates $\mathbf{q}$, along with the corresponding initial values. In addition, the user can define body-fixed *useful* PVs. The position and velocity of these PVs are calculable using the local frames of the rigid bodies via (2.1.2) and (2.2.6), respectively. The dynamical parameters of mechanism bodies, such as mass and the moment of inertia, need to be given as well. The user also specifies, in terms of $\mathbf{q}$ and/or $\dot{\mathbf{q}}$, quantities such

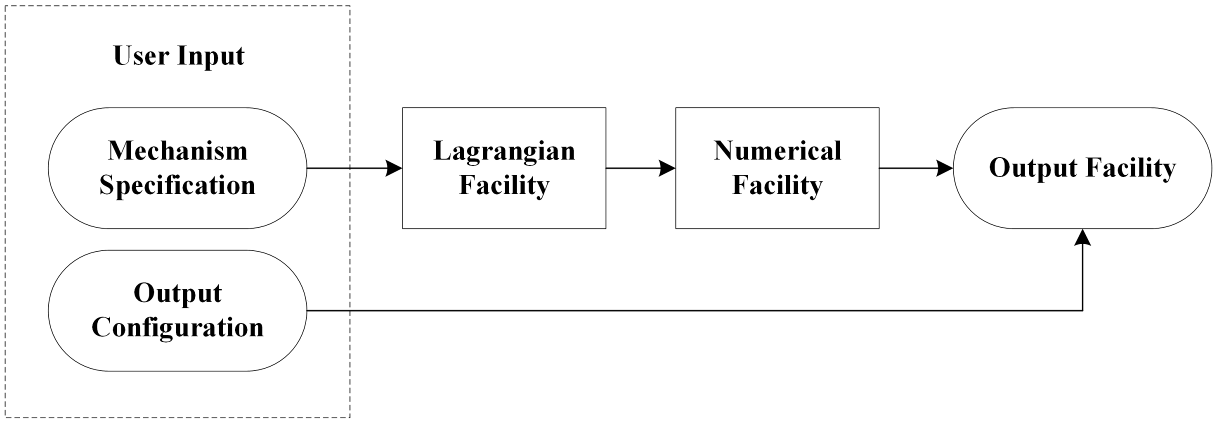


Figure 3.5: Simulation workflow.

as the kinetic and potential energies (accounting for the energy of all bodies and force elements), as well as Rayleigh's dissipation function, any external forces/torques. Lastly, the model requires a set of constraint equations, as outlined in Sections 1.2.1 and 2.5.

Apart from parameters that have constant values, the rest of the aforementioned quantities and relations are first constructed in a C++ code file using data types from the automatic differentiation package FADBAD++ [5], and then passed on to the Lagrangian facility of DAETS. For more details on the code implementation, see Appendix A.

3.2 DAETS and Its Lagrangian Facility

The DAETS solver accepts a general system of n equations in n state variables $x_j(t)$:

$$f_i(t,\ \text{the } x_j \text{ and derivatives of them}) = 0, \qquad i,j = 1,\dots,n. \tag{3.2.1}$$

From an encoding of the Lagrangian, constraint equations, etc., the Lagrangian facility applies the differentiations in (1.2.1) to build a DAE system of the form (3.2.1). More specifically, the time and partial derivatives d/dt, $\partial/\partial\mathbf{q}$ and $\partial/\partial\dot{\mathbf{q}}$ are constructed using FADBAD++ when converting to (3.2.1), thus avoiding the explicit derivations of the equations of motion by the user. The resulting

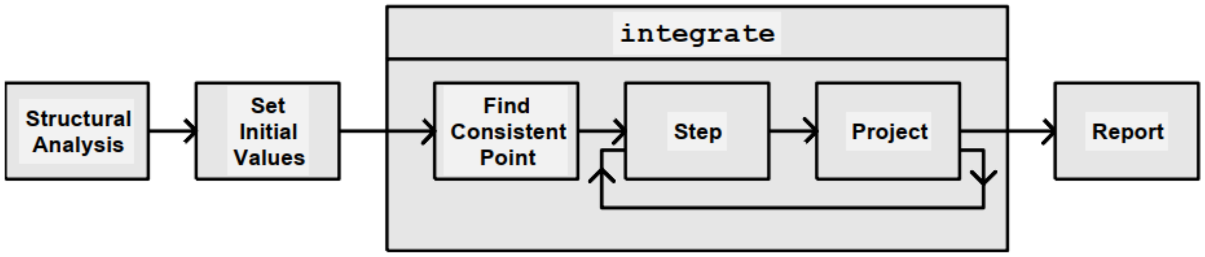


Figure 3.6: The workflow of DAETS' numerical engine, from [30, p. 2].

DAE has $n = n_q + n_c$ equations in state variables

$$\mathbf{x} = \left(q_1, \ldots, q_{n_q}, \lambda_1, \ldots, \lambda_{n_c}\right).$$

DAETS also supports adding user-defined variables governed by equations separate from (1.2.1) and (1.2.2) by manually expanding the state variables vector $\mathbf{x}$, which by default consists of the generalized coordinates and the Lagrange multipliers, in that order. Consider, for instance, the case of a ground-mounted motor rotating a rod with constant angular velocity ω. To track the total angle of rotation θ, one can append it to $\mathbf{x}$ and use the ODE

$$f_{n+1} = \frac{d\theta}{dt} - \omega = 0, \tag{3.2.2}$$

where n is the previous length of the state variables vector. Relation (3.2.2) is added as an extra equation to the system, and is solved alongside the primary DAE given by (1.2.1) and (1.2.2).

3.3 The Numerical Facility

DAETS follows the process illustrated in Figure 3.6 to integrate a system of DAEs numerically. At the start the structural analysis method of Pryce [37], effective for all DAEs arising from Lagrange's equations [37, Theorem 5.3], is applied to the DAE. This allows the solver to obtain the DAE index [6], DoF, quasilinearity status, and two non-negative integer arrays $\mathbf{d}$ and $\mathbf{c}$ with n components each, called the *equation offsets* and *variable offsets*, respectively. These arrays can be interpreted as follows.

If f_i is differentiated c_i times w.r.t. time, the system is converted to an ODE solvable for $x_i^{(d_j)}$—the *leading derivatives*—in terms of its lower derivatives. A system is quasilinear if the leading derivatives appear in a linear fashion in (3.2.1). The benefit of a quasilinear formulation for a mechanism is that it requires fewer IVs [30] compared to a fully nonlinear system with the same offset vectors.

Structural analysis also determines the variables and their derivatives that need to be specified as IVs to attain a *consistent* initial point [30], which might differ from the IVs defined by the user; in this event, the system prints an error, listing the required IVs. After being supplied with the necessary initial data, the solver tries to find a consistent initial point that is as close to that of the user input as possible via the optimization code IPOPT [47], by attempting to find a point in the configuration space that satisfies the constraint equations.

With a consistent point at hand, the integration procedure begins. DAETS uses an explicit Taylor Series numerical integration scheme [31,32,33] with fixed order and variable step-size selection [30, Section 2.1.3]. This implies that f_i need to be analytical along the solution path of the DAE for the integration to succeed, and thus cannot contain non-differentiable functions such as $\min, \max$, the step function, and so forth.

DAETS implements the EPUS (error per unit step) [42] method to control the local error at each time step during integration [30]. This approach leads to several desirable traits. One such trait is the property of tolerance proportionality, which means that the amount of error, measured w.r.t. true solution of the system (3.2.1), is proportional to the error tolerance specified by the user [42].

3.4 Output Facility

DAETS can report information on many aspects of the multibody system and its governing DAE before, during and after the simulation process. As outlined in Section 3.3, the system DoF, its offset vectors, signature matrix, etc. are obtained and displayed in a table format before integration starts. Output facility also has the following capabilities:

(a) reporting the time history of the numerical value of state variables and their derivatives (up to an order adjustable by the user);

(b) reporting the time history of any scalars and their derivatives as requested by the user, including the coordinates of useful PVs not in the state vector, user-defined quantities, etc;

(c) reporting outputs at a set of arbitrary time steps different from those used in the integration process—using high-order interpolation [30]—which offers valuable utility for validation studies and is used extensively in this thesis;

(d) setting the desired precision for output values, up to 16 decimal digits of accuracy.

Chapter 4

Case Study Template

This chapter presents the general structure of the case studies presented in Chapters 5 and 6. We explain the methodology used to model the multibody systems under study. In particular, we detail the process to select a suitable set of generalized coordinates. Moreover, we discuss the numerical results, validation technique, and verification process. Each case consists of three parts:

(a) problem specification, in which the physics of the mechanism and its operating conditions are described (Section 4.1);

(b) system derivation, where the procedure for selecting the generalized coordinates and local frames is given (Section 4.2);

(c) numerical results, which presents the time history of generalized coordinates, error plots, and a brief discussion, in which we interpret the obtained data and compare the accuracy of DAETS with reference models from Simulink (Section 4.3).

4.1 Problem Specification

We start each case study by describing the physical conditions that govern the motion of the mechanism of interest. This includes the number of rigid bodies, their arrangement relative to each other, the placement of joints and force transmission elements, external forces and torques, and so forth. Schematics are also provided to help visualize the problem. Geometrical data specific to each body, such as shape and dimensions, as well as dynamical data, including mass and moment of inertia, also belong here. If there are any springs or dampers in the system, when warranted, we describe them in a separate table. Note that this section only provides parameter symbols so that readers can understand the equations and the contribution of each system element to them. Actual parameter values are given in the numerical results section.

4.2 System Derivation

This section is dedicated to forming the generalized coordinates vector. We need to identify appropriate points and vectors to serve as local frame BPVs for each rigid body. In general, a body needs three BPVs in the 3D case and two BPVs in the 2D case. It is wise to choose points or directions of interest as BPVs. This will allow the user to obtain motion data for that point directly from $\mathbf{q}$, without the need for extra post-processing to compute position/velocity via a local frame from (2.1.2) or (2.2.6). Potential PVs of interest include the CoM, joint connection points, etc.

We also take advantage of the mechanism's kinematic and geometric relations to reduce the number of required coordinates. Fixed PVs with constant coordinates in the WF, such as the sliding direction vector of a prismatic joint connected to the ground, are a frequently occurring example. Using this vector as a BPV means fewer coordinates will be in $\mathbf{q}$. Another way of shortening $\mathbf{q}$ is to construct local frames using BPVs that are shared among two or more elements at joints. Revolute, spherical and prismatic joints enable this technique. For more detail refer to Section 2.5.

The next step is building the Lagrangian, Rayleigh's dissipation function (if applicable), and the constraint equations. The coordinates of body CoMs are either in $\mathbf{q}$ or computed as useful points so that their kinetic and potential energies are easily obtained. The same goes for spring/damper endpoints as well as joints. With all the aforementioned quantities at hand, we present the DoF table at the end of this section. This table summarizes the process of forming the local frames and is comprised of four columns for each body:

(a) **Bring in** lists the coordinates added to $\mathbf{q}$;

(b) **Assign** contains definitions for useful quantities, such as those computed from local frames or PVs that have already been determined;

(c) **Constrain** shows the constraint equations brought on by this body or the joints connected to it;

(d) **DoF** of a body is calculated by the difference between the number of its coordinates, and the number of its constraint equations.

4.3 Numerical Results

The results section is divided into three parts. In *Parameters*, the numerical value for constant parameters and IVs used for simulation are given. All numbers are in SI units, unless otherwise stated. *Results* presents superimposed time history curves of one or more Cartesian position/velocity components from DAETS and Simulink. In *Validation*, we provide error plots—measured between DAETS and reference solutions—over the entire simulation time. The source of reference data for validation is models of the analyzed mechanisms that we built inside the Simulink [26] environment of MATLAB [28], using the Simscape Multibody [27] package. The `ode113`, and `ode15s` (stiff) numerical solvers are used to simulate these models. Finally, *Discussion* presents an interpretation of the results provided in the preceding sections.

Computing solution differences and global error estimations at multiple tolerances is a standard method of assessing the accuracy of numerical solvers. The developers of DAETS have used this method in the past for the same purpose [29]. To stay consistent with previous works, we utilize the accuracy measuring techniques explained in this section. Obtaining analytical solutions for the problems presented in Chapters 5 and 6 is not possible as they are too complex. We also did not opt for simpler mechanisms because we needed to evaluate accuracy for more advanced problems that better reflected potential real-world usages of DAETS. As for experimental results, attaining them is time-consuming and costly. Moreover, results from experiments are prone to multiple forms of error. Consequently, they do not have the extremely high levels of accuracy we needed. As a whole, numerical results are the only source of reference data that are reasonably easy to obtain and sufficiently accurate for the purpose of our work.

To have a standardized method for establishing the accuracy of DAETS and comparing its results with Simulink, we compute two types of relative differences [29, Section 2.3] for the generalized coordinates across the entire simulation time span as follows:

- **Solution difference:** To obtain this quantity, the two solvers are run under four different absolute and relative error tolerances $10^{-6}, 10^{-8}, 10^{-10},$ and 10^{-12}. For a given tolerance, we measure the relative solution difference between DAETS and Simulink at each time step by

$$E = \frac{\left\| \mathbf{q}^{\text{DAETS}} - \mathbf{q}^{\text{Simulink}} \right\|_2}{\left\| \mathbf{q}^{\text{Simulink}} \right\|_2}. \tag{4.3.1}$$

- **Global error:** Here the Simulink model is run with absolute and relative error tolerances of 3×10^{-14}, which is close to the smallest value of 2.8×10^{-14} that MATLAB allows. This yields the *reference* solution, which is taken to be the best approximation available for the true solution of a given mechanism. On the other hand, DAETS' model is still run at the four tolerances mentioned above. With $\mathbf{q}$ as before, the global error estimate at each time step is defined as

$$E_{\text{global}} = \frac{\left\| \mathbf{q}^{\text{DAETS}} - \mathbf{q}_{\text{ref}}^{\text{Simulink}} \right\|_2}{\left\| \mathbf{q}_{\text{ref}}^{\text{Simulink}} \right\|_2}. \tag{4.3.2}$$

In case of both error types, Simulink is configured to generate its output at the time points chosen by DAETS.

The viability of our framework is assessed with regard to the two concepts of validation and verification. A valid model for a multibody system is one that correctly represents its kinematic and dynamical behavior. We demonstrate the validity of our natural coordinates models in each case by examining the difference between the initial consistent points computed by DAETS and Simulink, as well as the solution differences defined in (4.3.1). Note that the reference models are built in Simulink, an environment that uses different modeling and numerical schemes compared to DAETS. Considering this, we take the fact that DAETS and Simulink produce similar results with very small relative differences as sufficient evidence of validity. This means that both models correctly represent the same mechanism with the intended kinematic and dynamical properties.

The next step is verification, which deals with how accurate the results actually are. Here, we make use of the concept of global errors. Global error is defined as the relative difference between the numerical and true solutions of a problem [42]. With the validity of the reference models established, we take the best solution available from Simulink as our true (*i.e.*, reference) solution, and then estimate the global error by (4.3.2). For every case, we show that DAETS achieves a high degree of accuracy for all tolerances used. We also demonstrate that the global errors display the property of tolerance proportionality [42] in most mechanisms, a further indication of the robustness and accuracy of our approach.

Another piece of data we provide is the minimum number of significant digits in the solution vector components, denoted by SCD. This quantity is computed for each error tolerance using [25]

$$\textsc{scd} = -\log_{10}\big(\big\| \text{global error vector at simulation end time} \big\|_{\infty}\big),$$

where the global error vector is an array holding the relative global error for each individual solution component. This array is given by

$$\text{global error vector} = \Big(\frac{q_1^{\text{DAETS}} - q_1^{\text{Simulink}}}{q_1^{\text{Simulink}}}, \ldots, \frac{q_{n_q}^{\text{DAETS}} - q_{n_q}^{\text{Simulink}}}{q_{n_q}^{\text{Simulink}}}\Big),$$

where q_i^{DAETS} is the ith solution component for a given error tolerance, q_i^{Simulink} is Simulink's ith solution component obtained at the reference tolerance, and n_q is the length of the solution vector.

Finally, we measure the simulation times for DAETS and Simulink to compare their computational performance. Wall times are recorded on a desktop PC running Windows 10 Pro with a 64-bit Intel Core i7-3930K 3.2 GHz CPU and 12 GBs of RAM. DAETS runs on Windows Subsystem for Linux (WSL) [48]. For DAETS, timing is done via the Linux Bash command `time`, while for Simulink we use the `tic` and `toc` commands. The results for 2D and 3D problems are presented in

Sections 5.5 and 6.4, respectively.

Chapter 5

Two-dimensional Problems

5.1 N Four-bar Mechanism

5.1.1 Problem Specification

The N four-bar mechanism [17, 18] is a planar closed-loop linkage that is composed of N loops of rigid rods connected by revolute joints. Each two consecutive loops share a link, as shown in Figure 5.1. The constituent parts are $2N + 1$ rods.

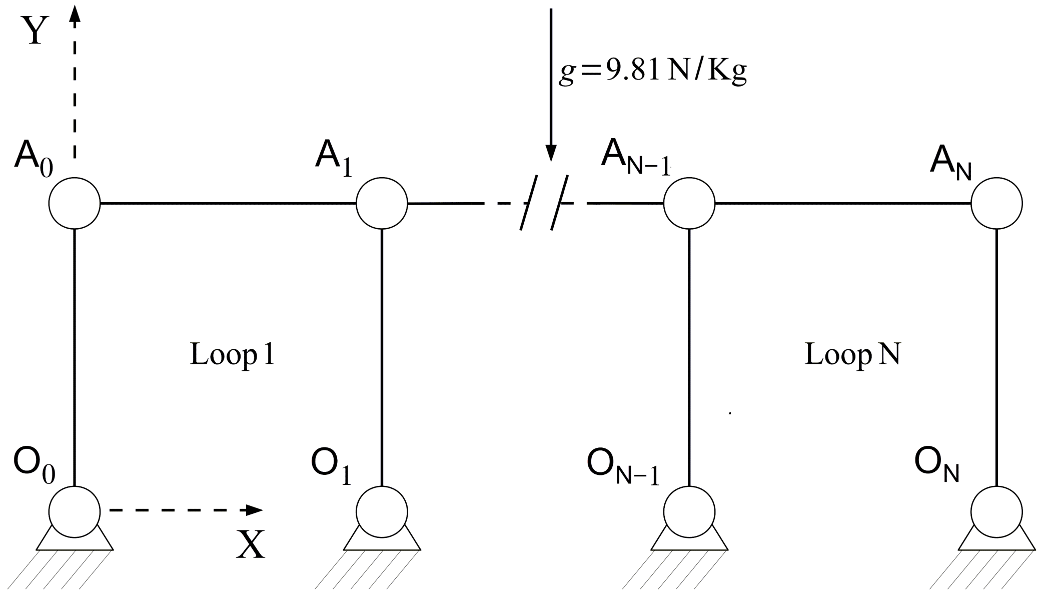


Figure 5.1: N four-bar mechanism—schematic.

Moving points are denoted by A_j and fixed points by O_j. Point O_0 is taken as the WF origin. In loop i, rod $\mathsf{O}_{i-1}\mathsf{A}_{i-1}$ is connected to the ground by a revolute joint at O_{i-1}, and to rod $\mathsf{A}_{i-1}\mathsf{A}_i$ via another revolute joint at A_{i-1}.

All rods have the same length ℓ and mass m, and are considered as thin rods (negligible cross-section area). Gravity is the only external force applied to the mechanism.

5.1.2 System Derivation

Coordinates. Each rod requires, at most, 2 points for the mechanism configuration to be uniquely determined. However, the rods connected to ground require only one, which we choose to be A_{i-1} in loop i. By adding A_i we can also track rod $\mathsf{A}_{i-1}\mathsf{A}_i$. As a result, all moving points need to be included in the generalized coordinates vector. The length of the vector will therefore be $2(N+1)$ and its components are:

$$\mathbf{q} = (\mathsf{A}_{0x}, \mathsf{A}_{0y}, \mathsf{A}_{1x}, \mathsf{A}_{1y} \dots, \mathsf{A}_{Nx}, \mathsf{A}_{Ny}).$$

Lagrangian. The total kinetic and potential energies of the system are obtained as follows

$$T = \sum_{i=1}^{N} \Big[\underbrace{\frac{1}{6} m \dot{\mathsf{A}}_{i-1}^2}_{\text{rod } \mathsf{O_{i-1}A_{i-1}}} + \underbrace{\frac{1}{6} m \big(\dot{\mathsf{A}}_{i-1}^2 + \dot{\mathsf{A}}_{i-1}\dot{\mathsf{A}}_i + \dot{\mathsf{A}}_i^2\big)}_{\text{rod } \mathsf{A_{i-1}A_i}} \Big] + \underbrace{\frac{1}{6} m \dot{\mathsf{A}}_N^2}_{\text{rod } \mathsf{O_N A_N}},$$

$$V = \sum_{i=1}^{N} \frac{1}{2} mg \Big[\underbrace{\mathsf{A}_{i-1_y}}_{\text{rod } \mathsf{O_{i-1}A_{i-1}}} + \underbrace{\big(\mathsf{A}_{i-1_y} + \mathsf{A}_{i_y}\big)}_{\text{rod } \mathsf{A_{i-1}A_i}} \Big] + \underbrace{\frac{1}{2} mg \mathsf{A}_{Ny}}_{\text{rod } \mathsf{O_N A_N}},$$

where the terms outside the summation represent the energy relations of the final, right-most rod in the last loop. The formula for the kinetic energy of the thin rods is given by [39].

Constraints. The explicit constraints consist only of rigidity relations for the rods. The restrictions imposed by the rods being connected by revolute joints are implicitly enforced via sharing a point between each two consecutive rods.

$$
\begin{aligned}
C_{2i-2} &= \left|\mathsf{A}_{i-1} - \mathsf{O}_{i-1}\right|^2 - \ell^2, \\
C_{2i-1} &= \left|\mathsf{A}_{i-1} - \mathsf{A}_{i}\right|^2 - \ell^2,
\end{aligned}
\qquad i = 1, 2, \ldots, N,
$$

$$
C_{2N} = \left|\mathsf{A}_{N} - \mathsf{O}_{N}\right|^2 - \ell^2.
$$

This leaves the mechanism with a DoF of $(2N+2) - (2N+1) = 1$. A summary of DoF calculations is given in Table 5.1. The subtraction in the **Total DoF** line is to avoid factoring in the DoF for shared points twice.

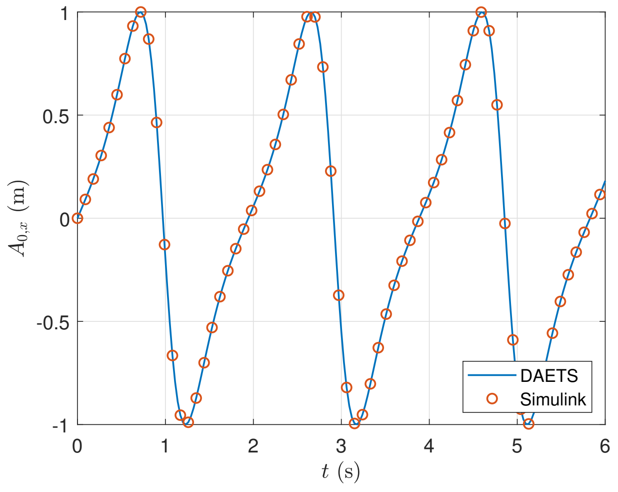
Table 5.1: N four-bar mechanism—DoF calculation for loop i.

Part	Bring in	Assign	Constrain	DoF
Rod $\mathsf{O}_{i-1}\mathsf{A}_{i-1}$ $i = 1, 2, \ldots, N$	A_{i-1x} A_{i-1y}		$0 = \left|\mathsf{A}_{i-1} - \mathsf{O}_{i-1}\right|^2 - \ell^2$	$2N - N = N$
Rod $\mathsf{A}_{i-1}\mathsf{A}_{i}$ $i = 1, 2, \ldots, N$	A_{ix} A_{iy}		$0 = \left|\mathsf{A}_{i-1} - \mathsf{A}_{i}\right|^2 - \ell^2$	$2N - N = N$
Rod $\mathsf{O}_{N}\mathsf{A}_{N}$	A_{Nx} A_{Ny}		$0 = \left|\mathsf{A}_{N} - \mathsf{O}_{N}\right|^2 - \ell^2$	$2 - 1 = 1$
Total DoF				$2N + 1 - 2N = 1$

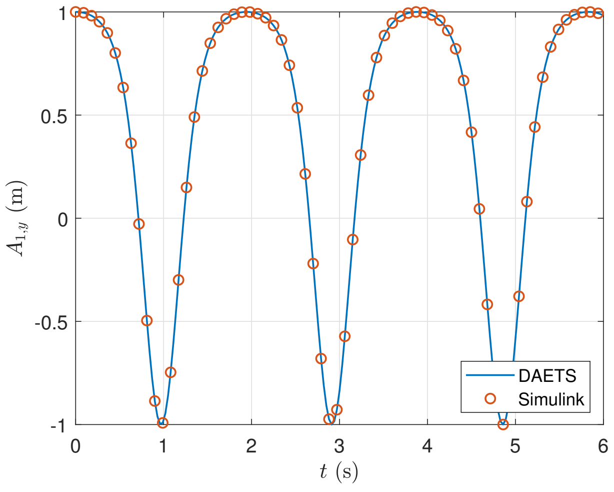
5.1.3 Numerical Results

Parameters. The initial state of the system is depicted in Figure 5.1, in which points A_i have a starting velocity of v_0 in the $+X$ direction. The results are obtained for a mechanism with two loops ($N = 2$), $m = 1$ kg, $\ell = 1$ m, and $v_0 = 1$ m/s.

Results. Time history of the global position coordinates $\mathsf{A}_{0,x}$ and $\mathsf{A}_{1,y}$ are presented in Figures 5.2a and 5.2b, respectively.



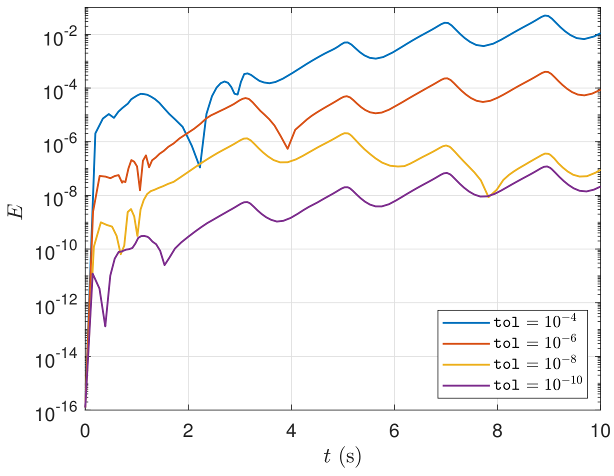
(a) $\mathsf{A}_{0,x}$.



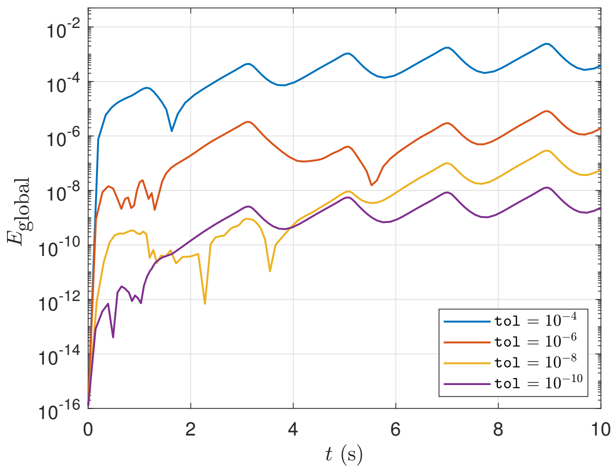
(b) $\mathsf{A}_{1,y}$.

Figure 5.2: N four-bar—position coordinates versus t.

Validation. Solution difference and global error plots are given in Figures 5.3a and 5.3b, respectively. Table 5.2 presents the minimum number of significant digits in the solution at $t = 10$ s.



(a) Solution difference.



(b) Global error.

Figure 5.3: N four-bar—difference between DAETS and Simulink solutions.

Table 5.2: N four-bar mechanism—minimum number of significant correct digits in the solution at the end of simulation.

Error tolerance	SCD
10^{-4}	2.70
10^{-6}	5.00
10^{-8}	6.52
10^{-10}	7.91

Discussion. Simulink cannot handle a tolerance of 10^{-12} or lower for this mechanism, potentially due to the occurrence of kinematic singularities when the mechanism goes into a fully horizontal configuration. This prompted us to use the tolerance values $10^{-4}, 10^{-6}, 10^{-8}$ and 10^{-10} for this problem.

The coordinate time histories of Figure 5.2 illustrate a good match between the solutions of DAETS and Simulink. The same is true for all other generalized position coordinates not shown, as evident from the relative solution differences in Figure 5.3a. The initial consistent points differ in the order of 10^{-16}, which means the kinematic properties of DAETS and Simulink models are computed accurately to precision. As a whole, the results are indicative of a good agreement between the two models, certifying that both are valid representations of the N four-bar mechanism.

Note that $\texttt{tol} = 5 \times 10^{-12}$ is used to obtain the reference solution. As shown in Figure 5.3b, the global error estimates are lower than relative differences at the same time steps. An interesting observation is that the errors for $\texttt{tol} = \texttt{10}^{-10}$ do not perform as well as expected and go above the errors for $\texttt{tol} = \texttt{10}^{-8}$ for $t < 4$. A potential cause of this is difficulties in maintaining kinematic constraints in lower tolerances due to the mechanism periodically going into singular configurations. Despite this, the global errors maintain acceptable values throughout simulation, which is demonstrative of an acceptable degree of accuracy for our natural coordinates model.

5.2 Slider-Crank Mechanism

5.2.1 Problem Specification

The slider-crank [17, 18] is a multibody system designed to transform angular motion into linear movement, and vice versa. This task is accomplished using an assembly of rigid bodies consisting of two thin rods, a sliding mass, and the ground link.

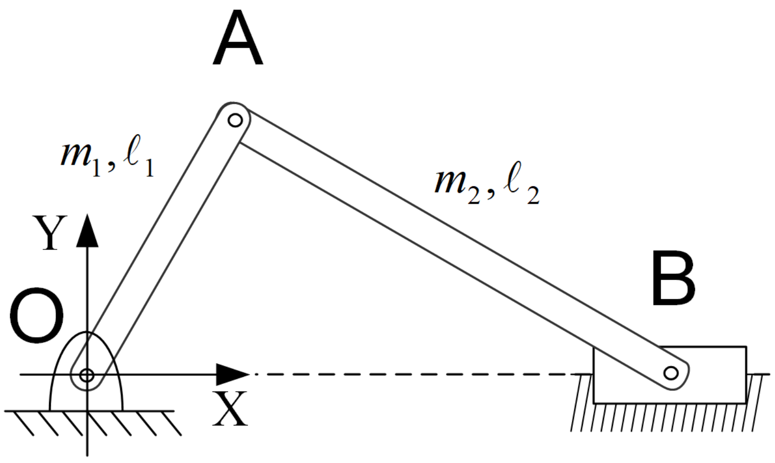


Figure 5.4: Slider-crank mechanism—schematic.

As depicted in Figure 5.4, the WF origin is placed at O. Rod OA is attached to the origin via a revolute joint. Rod AB is connected to both OA and the slider (B) by revolute joints as well, while the slider is linked to the ground using a prismatic joint. As a result, point B can only move horizontally. The slider is considered a point mass. Table 5.3 presents the physical parameters of the system.

Table 5.3: Slider-crank mechanism—system parameters.

Part	Mass	Length
Rod OA	m_1	ℓ_1
Rod AB	m_2	ℓ_2
Slider	0	

5.2.2 System Derivation

Coordinates. The two rods and the slider move in the global xy plane, necessitating tracking two points per body to uniquely define the system configuration at any time. However, point A is shared between the two rods, and so is B between the second rod and the slider. Also, the slider is a point mass, and its tracking requires a single coordinate, namely B_x. As a result, the generalized coordinates vector is obtained as:

$$\mathbf{q} = (\mathsf{A}_x, \mathsf{A}_y, \mathsf{B}_x).$$

Lagrangian. The total kinetic and potential energy terms of the system are obtained as

$$\begin{aligned} T &= \underbrace{\frac{1}{6} m_1 \dot{\mathsf{A}}^2}_{\text{rod 1}} + \underbrace{\frac{1}{6} m_2 \big(\dot{\mathsf{A}}^2 + \dot{\mathsf{A}}\dot{\mathsf{B}} + \dot{\mathsf{B}}^2\big)}_{\text{rod 2}}, \\ V &= \frac{1}{2} g \big[\underbrace{m_1 \mathsf{A}_y}_{\text{rod 1}} + \underbrace{m_2 \big(\mathsf{A}_y + \mathsf{B}_y\big)}_{\text{rod 2}}\big], \end{aligned}$$

where $\mathsf{B}_y = 0$, but it is not omitted in the formula for readability.

Constraints. There are a total of two constraint equations acting on the system that ensure the rigidity of rods OA and AB.

$$\begin{aligned} C_1 &= \big|\mathsf{O} - \mathsf{A}\big|^2 - \ell_1^2, \\ C_2 &= \big|\mathsf{B} - \mathsf{A}\big|^2 - \ell_2^2. \end{aligned}$$

A summary of DoF calculations are presented in Table 5.4.

Table 5.4: Slider-crank mechanism—DoF calculation.

Part	Bring in	Assign	Constrain	DoF
Rod OA	A_x A_y		$0 = \lvert\mathsf{O} - \mathsf{A}\rvert^2 - \ell_1^2$	$2 - 1 = 1$
Rod AB	B_x		$0 = \lvert\mathsf{B} - \mathsf{A}\rvert^2 - \ell_2^2$	$1 - 1 = 0$
Slider				
Total DoF				1

5.2.3 Numerical Results

Parameters. Initially, the system starts moving under the effect of gravity, with the slider having an initial velocity of -4 m/s in the x-direction. All other parts are initially at rest. Numerical values for system parameters and IVs given to DAETS are provided in Tables 5.5 and 5.6, respectively.

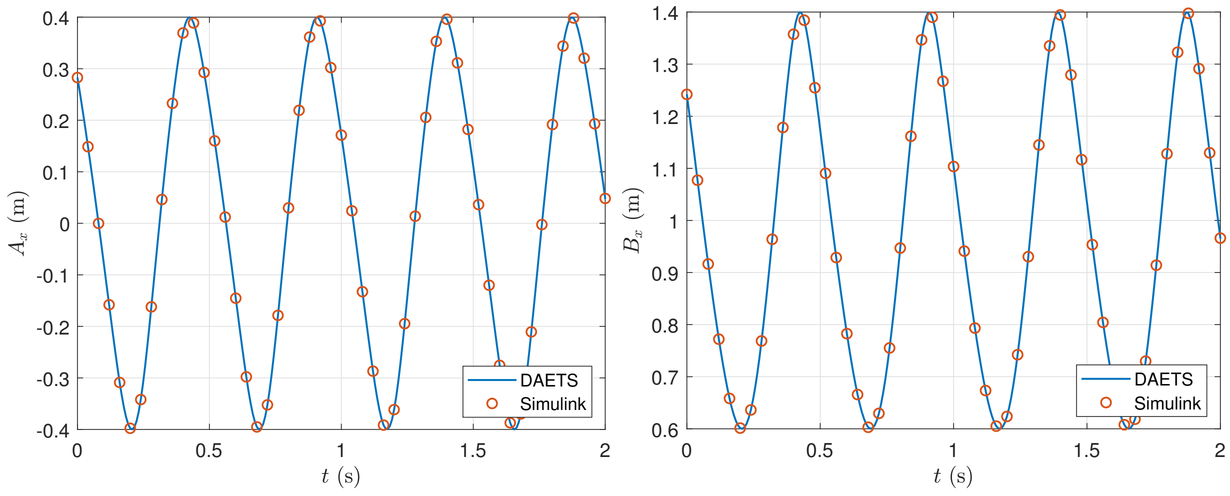
Table 5.5: Slider-crank mechanism—simulation parameters.

Part	Mass	Length	Width
Rod OA	1	0.4	0.04
Rod AB	1	1	0.04
Slider	0		

Table 5.6: Slider-crank mechanism—initial values.

Variable	x	$\dot{x}$
A_x	0.28	0
A_y	0.28	0
B_x	1.24	-4

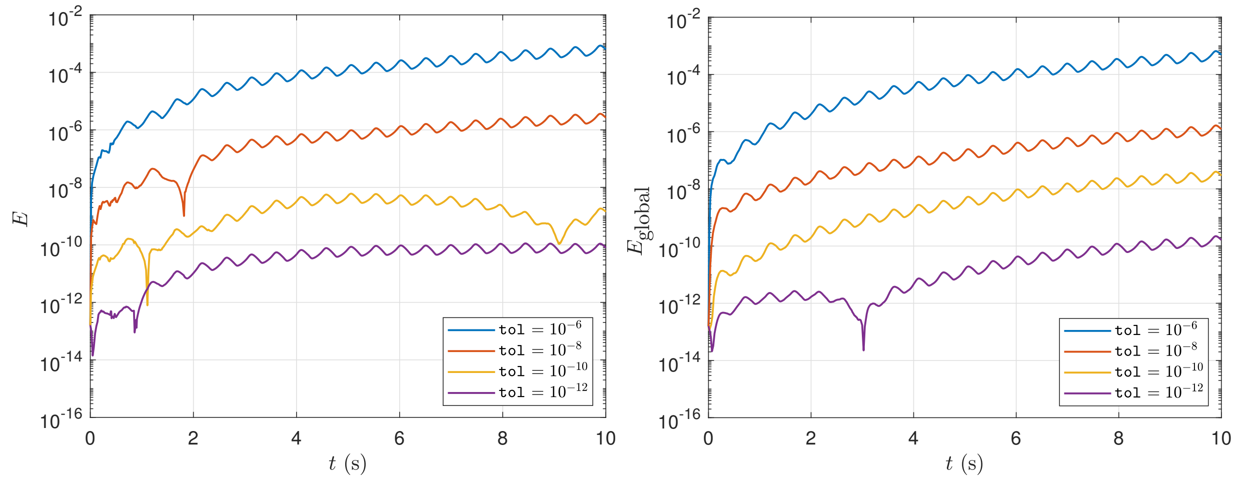
Results. Figures 5.5a and 5.5b present the time history of the global x-coordinates for points A and B.



(a) A_x. (b) B_x.

Figure 5.5: Slider-crank—x-coordinates of A and B versus t.

Validation. Solution differences and global error plots are given in Figures 5.6a and 5.6b, respectively. Table 5.7 presents the minimum number of significant correct digits in DAETS' solution at the end of simulation interval.



(a) Solution difference. (b) Global error.

Figure 5.6: Slider-crank—difference between DAETS and Simulink solutions.

Discussion. Figure 5.6a illustrates that the solution differences start at a value in the order of 10^{-13} regardless of error tolerance, indicating that DAETS manages to find a consistent point

Table 5.7: Slider-crank—minimum number of significant correct digits in the solution at the end of simulation.

Error tolerance	SCD
10^{-6}	2.57
10^{-8}	5.16
10^{-10}	6.79
10^{-12}	9.04

highly similar to the initial values that Simulink calculates. Therefore, it is clear that DAETS and Simulink's models have similar kinematic properties. After the initial upticks for $t < 2$, solution differences display a weakening upside trend—*i.e.* the growth rate decreases as time goes on. Hence, we can say that the difference between DAETS and Simulink solutions stays within a reasonable bound. The growth factor of differences is about 10^5 for `tol` $= 10^{-6}, 10^{-8}$, while for `tol` $= 10^{-10}$ and 10^{-12} the growth rate is about 10^4. Also, the two stricter tolerances show a generally flattened trend after $t = 6$. This demonstrates that using more stringent tolerances is beneficial for lessening the effect of error accumulation over time. We can conclude from the above information that there is a high degree of agreement between the two solvers, and that both DAETS and Simulink have valid models.

By examining the global error estimates in Figure 5.6b we can see that the initial error values are in the order of 10^{-13}. Also, the trend is similar to that of solution differences. Moreover, error values are reduced proportional to the reduction in error tolerances at the same time steps. Based on the data in Table 5.7, the maximum relative error observed at $t = 10$ s is about 2.69×10^{-3} for `tol` $= 10^{-6}$, which is indicative of a good match between the results of Simulink and DAETS. As a whole, the presented data verifies the accuracy of our natural coordinates model.

5.3 Copy Machine

5.3.1 Problem Specification

The copy machine is a planar mechanism that simulates a simplified version of a paper copying device. It is comprised of the parts listed below:

- a rod
- a square box
- two springs
- a damper

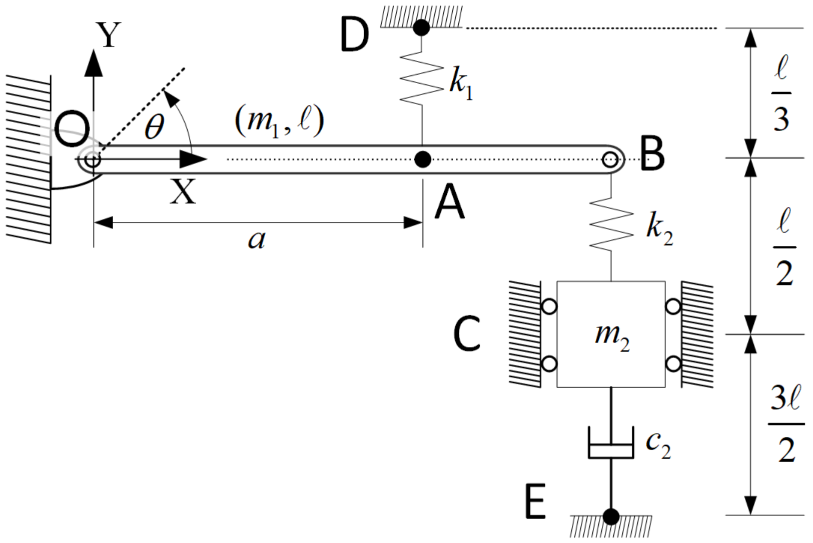


Figure 5.7: Copy machine—schematic.

As Figure 5.7 illustrates, the world frame origin is at $\mathsf{O} = (0, 0, 0)$. Rod OB is connected to the origin by a revolute joint. A spring with stiffness k_1 links point A on the rod to the fixed point D, with $a = 0.7\ell$, where ℓ is the rod length. The box, which is treated as a point mass and is restricted to vertical movement, is designated by C and is connected to the ground at point E via a damper, and to point B by the spring k_2. The angle between the rod and the WF x-axis is denoted by θ. System parameters are described in Table 5.8.

Table 5.8: Copy machine—system parameters.

Part	Mass	Length	Width	Height
Rod	m_1	ℓ	w	h
Box	m_2			
	Mass	Nat. length	Stiffness	Damp. coeff.
Spring AD	0	$\ell/3$	k_1	0
Spring BC	0	$\ell/2$	k_2	0
Damper CE	0	$3\ell/2$	0	c_2

5.3.2 System Derivation

Coordinates. The copy machine is a planar linkage comprised of two rigid parts, and it needs three generalized coordinates to fully define the position and velocity of its components. The following describes how these coordinates are chosen:

Rod Since it is connected at one end to the fixed point O, the rod only requires one more point to define its local frame, which is taken to be the endpoint B. Therefore, the local frame is built as OB. Furthermore, the angle θ is obtained [29] from

$$\dot{\theta} = \frac{\mathsf{B}_x\dot{\mathsf{B}}_y - \mathsf{B}_y\dot{\mathsf{B}}_x}{\ell^2}. \tag{5.3.1}$$

Box As stated in the previous subsection, the box is treated as a point mass named C. Given that C_x is fixed and equal to ℓ, this part only adds C_y to the $\mathbf{q}$ vector.

The $\mathbf{q}$ vector is thus given by:

$$\mathbf{q} = (\mathsf{B}_x, \mathsf{B}_y, \mathsf{C}_y).$$

Lagrangian. The kinetic and potential energy terms are given by

$$T = \frac{1}{2}m_1\dot{\mathsf{B}}^2 + \frac{1}{2}I_1\dot{\theta}^2 + \frac{1}{2}m_2\dot{\mathsf{C}}^2, \tag{5.3.2}$$

$$V = \frac{1}{2}m_1 g\mathsf{B}_y + m_2 g\mathsf{C}_y + \frac{1}{2}k_1\big(|\mathsf{A}-\mathsf{D}| - \frac{\ell}{3}\big)^2 + \frac{1}{2}k_2\big(|\mathsf{B}-\mathsf{C}| - \frac{\ell}{2}\big)^2; \tag{5.3.3}$$

where $I_1 = \frac{1}{12}m_1(\ell^2 + h^2)$ is the moment of inertia of the rod about WF origin.

Dissipation. The Rayleigh function for modeling the dampers is given by

$$\mathcal{R} = \frac{1}{2}c_2\dot{\mathsf{C}}^2. \tag{5.3.4}$$

Constraints. The only constraint equation is the rigidity of rod OB:

$$C = |\mathsf{B}-\mathsf{O}|^2 - \ell^2. \tag{5.3.5}$$

Table 5.9 has a summary of the entire DoF calculation process for this problem.

Table 5.9: Copy machine—DoF calculation.

Part	Bring in	Assign	Constrain	DoF
Rod	B_x B_y		$0 = \|\mathsf{B}-\mathsf{O}\|^2 - \ell^2$	$2-1=1$
Box	C_y			$1-0=1$
Total DoF				2

5.3.3 Numerical Results

Parameters. Numerical values for system parameters and IVs given to DAETS are provided in Tables 5.10 and 5.11, respectively.

Table 5.10: Copy machine—simulation parameters.

Part	Mass	Length	Width	Height
Rod	1	1	0.02	0.02
Box	1			
	Mass	Nat. length	Stiffness	Damp. coeff.
Spring AD	0	0.33	100	0
Spring BC	0	0.5	100	0
Damper CE	0	1.5	0	50

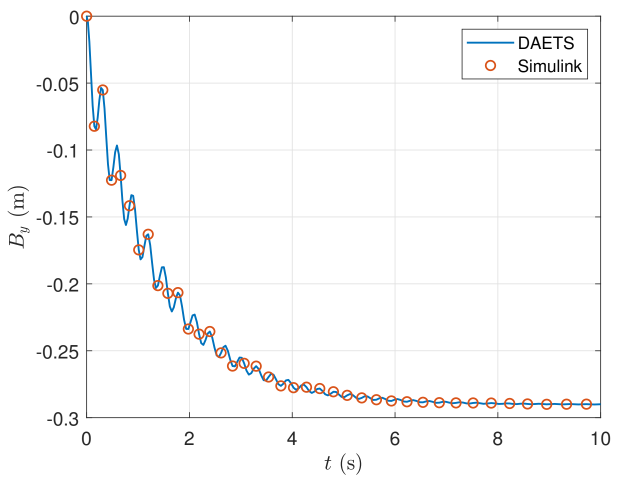
Table 5.11: Copy machine—initial values.

Variable	x	$\dot{x}$
B_x	1	0
B_y	0	0
C_y	−0.5	0

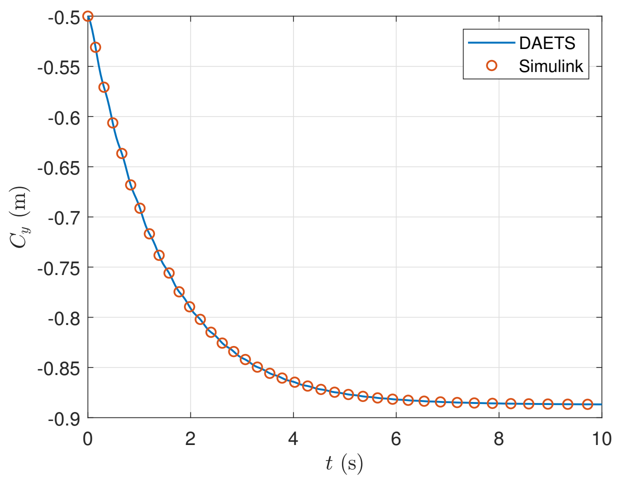
Results. Figures 5.8b and 5.8a display the time history of C_y and B_y, respectively.

Validation. Solution difference and global error plots are presented in Figures 5.9a and 5.9b, respectively.

Discussion. There is great agreement between Simulink and DAETS's solutions displayed in Figure 5.8. The solution differences in Figure 5.9a start with a value in the order of $10^{-9}, 10^{-11}, 10^{-13}$, and 10^{-15} for the highest to lowest tolerance, respectively. This shows that DAETS computes an initial consistent point that is sufficiently close to the initial state provided as input to Simulink. Given that the initial consistent point for a mechanism is subject to its constraints and physical parameters, we can say that both models have the same kinematic properties.

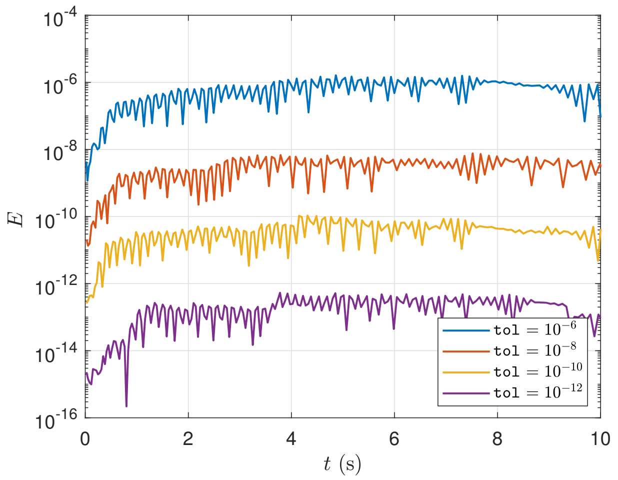


(a) B_y.

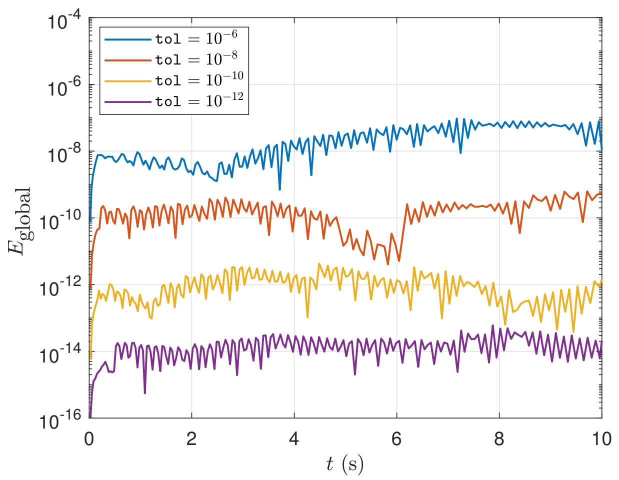


(b) C_y.

Figure 5.8: Copy machine—y-coordinates of B and C versus t.



(a) Solution difference.



(b) Global error.

Figure 5.9: Copy machine—difference between Simulink and DAETS solutions.

The difference curves stop growing after $t = 4$ as system motion dies down thanks to the damper. As a result, difference values stay below the error tolerances most of the time, indicating that both DAETS and Simulink are producing very similar results. We take this as sufficient proof that our natural coordinates model is valid.

By examining Figure 5.9b one can observe tolerance proportionality during most of the simulation period, with the exception of very short intervals, such as $5 < t < 6$ where the global errors for $\texttt{tol} = 10^{-8}$ are much lower than expected. Furthermore, the data in Table 5.12 shows a high number of significant digits given the error tolerance used. We can conclude that the natural

Table 5.12: Copy machine—minimum number of significant correct digits in the solution at the end of simulation.

Error tolerance	SCD
10^{-6}	7.42
10^{-8}	8.57
10^{-10}	11.19
10^{-12}	13.04

coordinates model gives an accurate solution for all tolerances.

5.4 McPherson Strut Suspension System

5.4.1 Problem Specification

The McPherson strut suspension is a popular linkage that is widely used as the front suspension in modern vehicles. A schematic of the model studied here [1] is depicted in Figure 5.10, showing the following components:

- car body (inertial frame)
- unsprung mass
- control arm
- unsprung mass
- actuator plate

Each of these parts represents an amalgamation of the components of a real McPherson strut suspension. The movement of the parts is planar, leading to a 2D model. Furthermore, all bodies are rigid and their deformations, along with friction effects, are neglected.

The unsprung mass is connected to the car body by a prismatic joint (called the slider), restricting it to vertical motion in the global Y direction. The same is true for the actuator plate. The control arm is a link connected to the sprung mass via a revolute joint at point A, and by another revolute joint to the unsprung mass at B. The unsprung mass has three points of interest. First, it is joined to the sprung mass via the primary spring-damper that connects points Q and P, the exact position of which is defined below. Second, it is linked to the actuator by the tire spring-damper with its endpoints at U and R. Finally, it is connected to the control arm at B.

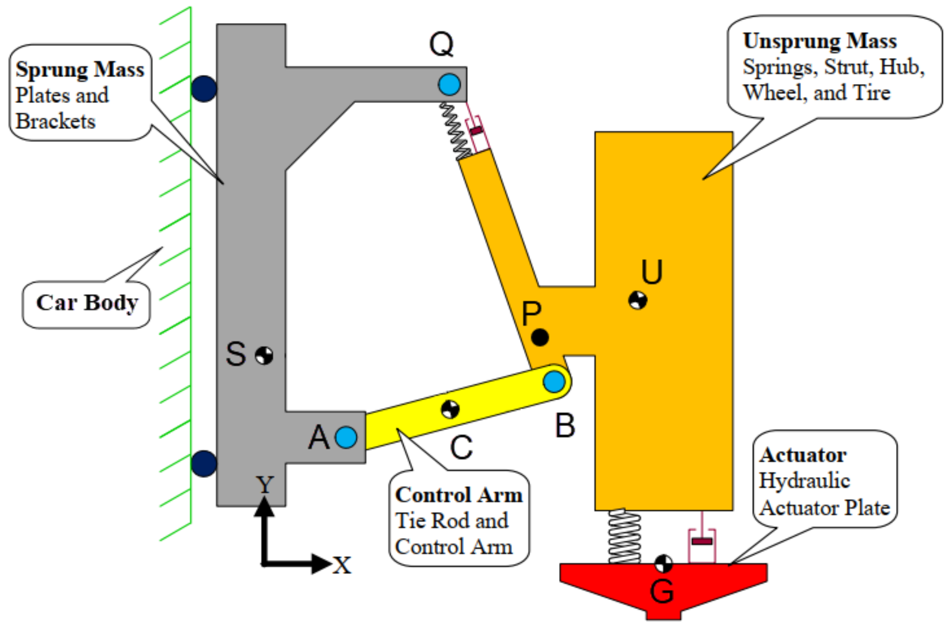


Figure 5.10: McPherson suspension—schematic [1, p. 49].

Figure 5.11 shows the suspension schematic along with all the basic ($\mathsf{A}, \mathsf{B}, \mathsf{P}$, and G) and useful ($\mathsf{S}, \mathsf{Q}, \mathsf{C}$, and U) PVs marked. C, S and U denote the centers of gravity for the control arm, sprung mass, and unsprung mass, respectively. G marks the connection spot of the tire spring to the actuator, while P denotes where the primary spring meets the unsprung mass. P is positioned on the unsprung mass such that PU is perpendicular to QB (the *strut arm*).

The system parameters are presented in Table 5.13, and Figure 5.12 shows the geometric constants.

Table 5.13: McPherson suspension—system parameters.

Part	Mass	MoI
Sprung mass	m_s	I_s
Control arm	m_c	I_c
Unsprung mass	m_u	I_u
Actuator plate	0	0

	Mass	Nat. length	Stiffness	Damp. coeff.
Primary spring-damper	0	d_p	k_p	c_p
Tire spring-damper	0	d_t	k_t	c_t

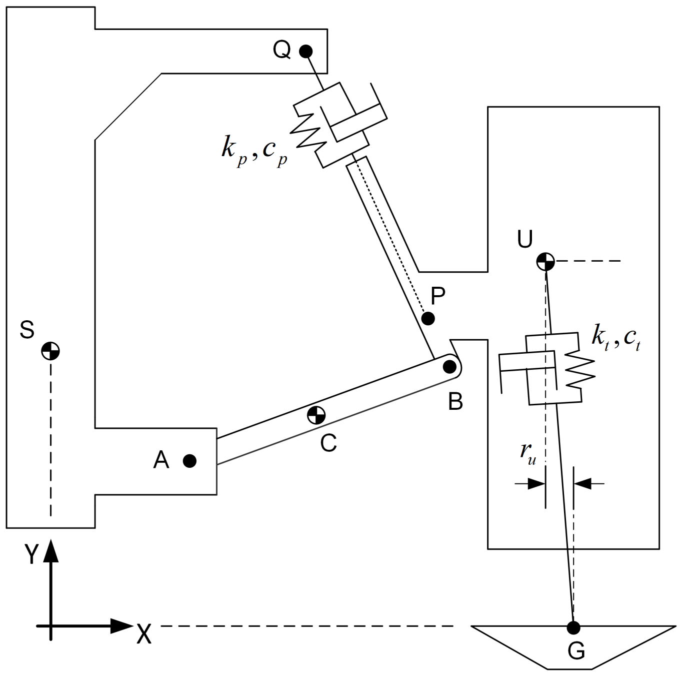


Figure 5.11: McPherson suspension—schematic with important PVs.

5.4.2 System Derivation

Coordinates. The system consists of five rigid bodies with the car body acting as the inertial frame. This leaves four bodies moving in a 2D plane, meaning that eight coordinates are needed to fully define the system configuration at any point in time. However, this number can be reduced as described below.

Sprung mass The slider connecting this object to the car body prevents horizontal as well as rotational motion, meaning that the x position of all body points are constant. Hence, A_y is the only coordinate from the sprung mass added to the $\mathbf{q}$ vector. No local frames are created for the sprung mass.

Control arm Point A is already accounted for; therefore B is chosen as the second point tracked on this body, enabling us to form the body-fixed frame AB. The spin angular velocity for the control arm, ω_c, is computed about the z-axis.

Unsprung mass This part adds point P to vector $\mathbf{q}$. With B already in the $\mathbf{q}$ vector, a body-fixed

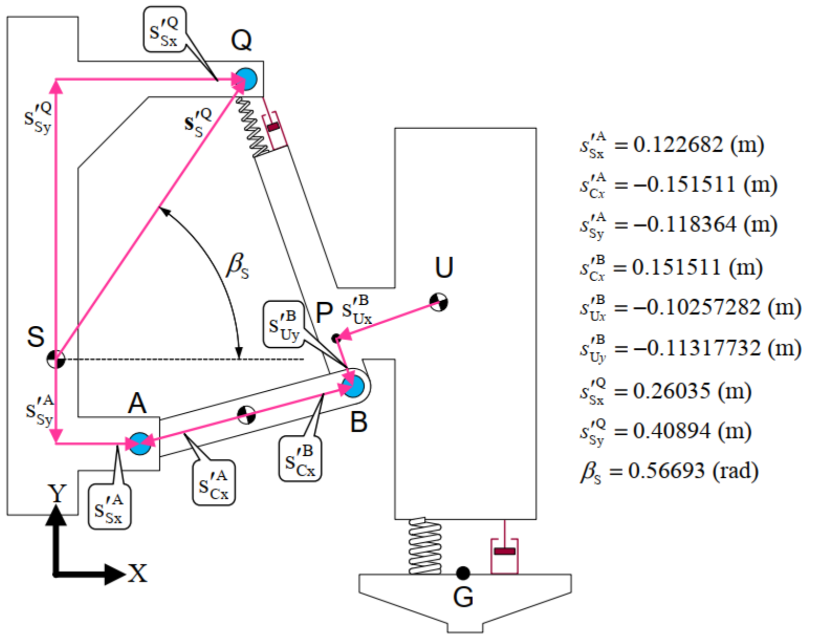

s'^{Q}_{Sx}
Q
s'^{Q}_{Sy}
s'^{Q}_{S}
β_S
U
s'^{B}_{Uy}
P
s'^{B}_{Ux}
S
A
B
s'^{A}_{Sy}
s'^{B}_{Cx}
s'^{A}_{Cx}
s'^{A}_{Sx}
Y
X
G
$s'^{A}_{Sx} = 0.122682$ (m)
$s'^{A}_{Cx} = -0.151511$ (m)
$s'^{A}_{Sy} = -0.118364$ (m)
$s'^{B}_{Cx} = 0.151511$ (m)
$s'^{B}_{Ux} = -0.10257282$ (m)
$s'^{B}_{Uy} = -0.11317732$ (m)
$s'^{Q}_{Sx} = 0.26035$ (m)
$s'^{Q}_{Sy} = 0.40894$ (m)
$\beta_S = 0.56693$ (rad)

Figure 5.12: McPherson suspension—geometric constants [1, p. 77].

frame can be formed with its origin at P and x-axis towards PB. From this frame ω_u, as well as the position and velocity of U, are calculable, as outlined in Table 5.14.

Actuator plate Appending G_y to the generalized coordinates vector is sufficient for tracking this body as it is constrained to vertical motion only. The movements of the actuator represent the road shape profile and determine the force it exerts on the tire, which the suspension must absorb rather than transmit to the passengers cabin. A sinusoidal road shape with an amplitude of G_{amp} and a period of G_{period} is considered so that

$$\mathsf{G}_y = G_{\text{amp}} \sin\left(\frac{2\pi}{G_{\text{period}}} t\right). \tag{5.4.1}$$

The generalized coordinates vector is thus obtained with six components:

$$\mathbf{q} = (\mathsf{A}_y, \mathsf{B}_x, \mathsf{B}_y, \mathsf{P}_x, \mathsf{P}_y, \mathsf{G}_y).$$

Lagrangian. Kinetic and potential energy terms are defined by

$$T = \underbrace{\frac{1}{2} m_\text{s} \dot{\mathsf{S}}^2}_{\text{sprung mass}} + \underbrace{\frac{1}{2}\left(m_\text{c}\dot{\mathsf{C}}^2 + I_\text{c}\omega_c^2\right)}_{\text{control arm}} + \underbrace{\frac{1}{2}\left(m_\text{u}\dot{\mathsf{U}}^2 + I_\text{u}\omega_\text{u}^2\right)}_{\text{unsprung mass}},$$

$$V = g(\underbrace{m_\text{s}\mathsf{S}_y}_{\text{sprung mass}} + \underbrace{m_\text{c}\mathsf{C}_y}_{\text{control arm}} + \underbrace{m_\text{u}\mathsf{U}_y}_{\text{unsprung mass}}) + \frac{1}{2}\Big[\underbrace{k_\text{p}\left(\left|\mathsf{P}-\mathsf{Q}\right| - d_\text{p}\right)^2}_{\text{primary spring}} + \underbrace{k_\text{t}(\left|\mathsf{U}-\mathsf{G}\right| - d_\text{t})^2}_{\text{tire spring}}\Big].$$

Dissipation. The primary and tire dampers are modeled by the Rayleigh function as

$$\mathcal{R} = \underbrace{\frac{1}{2}c_\text{p}\left[(\dot{\mathsf{P}} - \dot{\mathsf{Q}})\cdot u_{\mathsf{PQ}}\right]^2}_{\text{primary damper}} + \underbrace{\frac{1}{2}c_\text{t}\left|\dot{\mathsf{U}} - \dot{\mathsf{G}}\right|^2}_{\text{tire damper}},$$

where u_{PQ} is a unit vector parallel to the strut arm given by

$$u_{\mathsf{PQ}} = \frac{\mathsf{P}-\mathsf{Q}}{\left|\mathsf{P}-\mathsf{Q}\right|}.$$

Constraints. As explained above, the constraints caused by the slider joints connected to the sprung mass and actuator plate are accounted for by keeping the x position of their respective bodies constant. The revolute joint constraints are brought in implicitly by sharing points A, B and Q between bodies. This leaves the rigidity relations of the control arm and unsprung mass (the sprung mass requires none as it is represented by a single point in the $\mathbf{q}$ vector), the assumed perpendicularity of PU to the strut arm, as well as the prescribed motion of the actuator (per Equation (5.4.1)), as the explicit constraint equations:

$$C_1 = |\mathsf{A}-\mathsf{B}|^2 - \ell_{\mathsf{AB}}^2,$$

$$C_2 = |\mathsf{P}-\mathsf{B}|^2 - \ell_{\mathsf{PB}}^2,$$

$$C_3 = (\mathsf{P}-\mathsf{B})\cdot(\mathsf{P}-\mathsf{U}),$$

$$C_4 = \mathsf{G}_y - G_\text{amp}\sin\left(\frac{2\pi}{G_\text{period}}t\right),$$

where $\ell_{\mathsf{AB}} = s'^{\mathrm{A}}_{\mathrm{Cx}} + s'^{\mathrm{B}}_{\mathrm{Cx}}$ and $\ell_{\mathsf{PB}} = s'^{\mathrm{B}}_{\mathrm{Uy}}$ from Figure 5.12. A summary of the constraints and its DoF computation are presented in Table 5.14.

The relative coordinates model used in [1], where the problem definition was taken from, uses a generalized coordinates vector with 10 components, including two angles, as well as eight constraint equations, leading to a system of size 18. Our model, in comparison, has only six generalized coordinates and four constraint equations, resulting in a DAE system of size 10.

Table 5.14: McPherson suspension—DoF calculation.

Part	Bring in	Assign	Constrain	DoF
Sprung mass	A_y			$1-0=1$
Control arm	B_x B_y		$0=\left\|\mathsf{A}-\mathsf{B}\right\|^2-\ell_{\mathsf{AB}}^2$	$2-1=1$
Unsprung mass	P_x P_y	$\mathsf{U}=\mathsf{P}+\mathbf{Q}_{\mathrm{u}}\mathsf{U}_0$ $\mathsf{U}_0=\big(0,\ell_{\mathsf{PU}}\big)$	$0=\left\|\mathsf{P}-\mathsf{B}\right\|^2-\ell_{\mathsf{PB}}^2$ $0=(\mathsf{P}-\mathsf{B})\cdot(\mathsf{P}-\mathsf{U})$	$2-2=0$
Actuator plate	G_y		$0=\mathsf{G}_y-G_{\mathrm{amp}}\sin(\frac{2\pi}{G_{\mathrm{period}}}t)$	$1-1=0$
Total DoF				2

5.4.3 Numerical Results

Parameters. Table 5.15 contains numerical values for the system parameters. Initial positions and orientation angles for the centers of gravity are given in Table 5.16, where C_ϕ is measured from the global X direction to AB, and U_ϕ is the angle between the global x-axis and PU. Also, values for G_{amp} and G_{period} are set to 4×10^{-3} m and 0.2 s, respectively.

Table 5.15: McPherson suspension—simulation parameters.

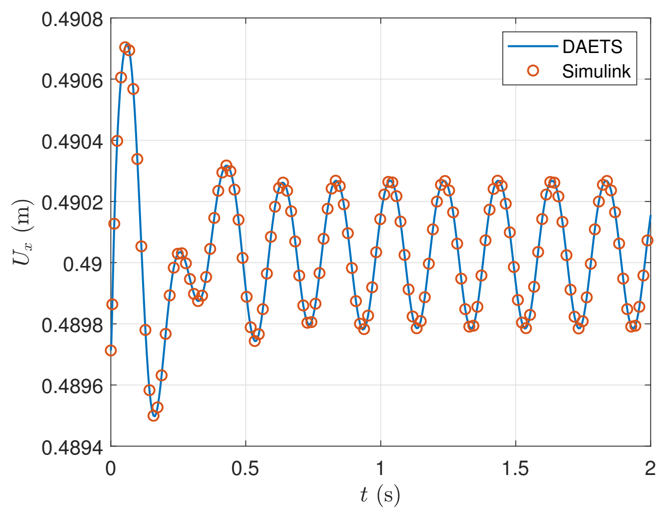
Part	Mass		MoI	
Sprung mass	240		17.85	
Control arm	3		0.03	
Unsprung mass	40		1.58	
Actuator plate	0		0	
	Mass	Nat. length	Stiffness	Damp. coeff.
Primary spring-damper	0	0.5034	1.4×10^4	6×10^3
Tire spring-damper	0	0.3	4×10^5	10^4

Table 5.16: McPherson suspension—initial values.

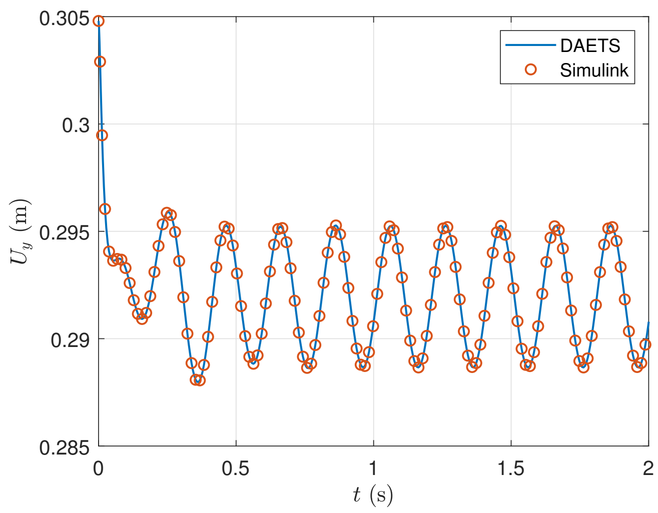
Variable	x	$\dot{x}$
A_y	0.17	0
B_x	0.43	0
B_y	0.17	0
P_x	0.39	0
P_y	0.27	0
U_ϕ	0.30	0
G_y	0	0.13

Results. The time histories of the position coordinates of U are depicted in Figure 5.13. Position plots are given only up to two seconds because of the periodic behavior of the system.

Validation. Solution differences and global errors are illustrated in Figures 5.14a and 5.14b, respectively. Table 5.17 provides the minimum number of significant correct digits in the solution at the end of simulation.

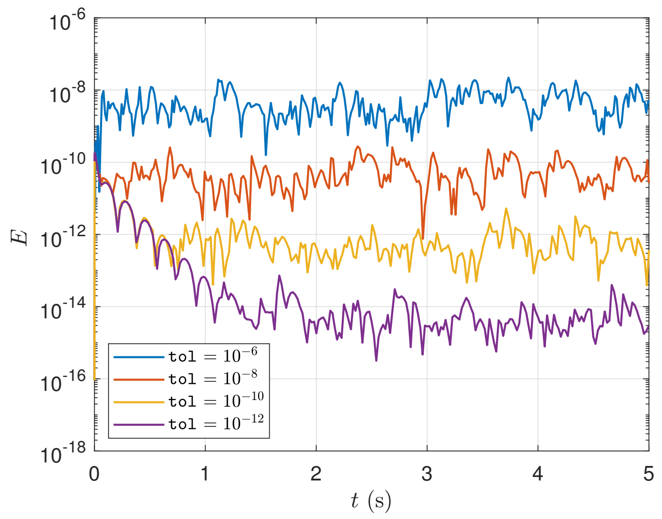


(a) U_x.

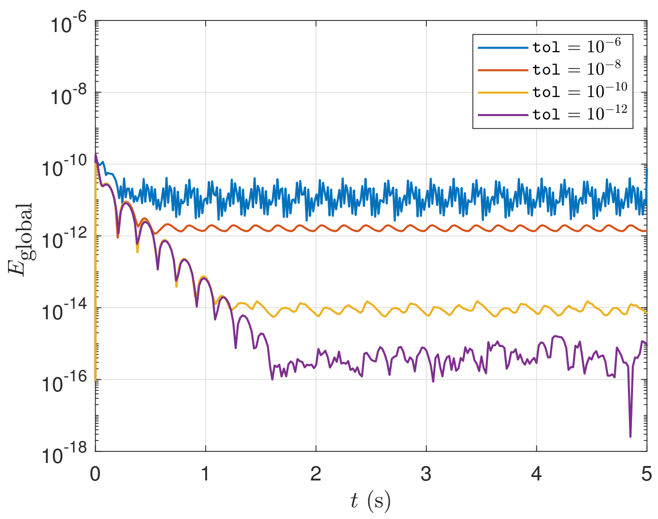


(b) U_y.

Figure 5.13: McPherson suspension—position coordinates of U versus t.



(a) Solution difference.



(b) Global error.

Figure 5.14: McPherson suspension—difference between DAETS and Simulink solutions.

Table 5.17: McPherson suspension—minimum number of significant correct digits in the solution at the end of simulation.

Error tolerance	SCD
10^{-6}	10.62
10^{-8}	11.48
10^{-10}	13.65
10^{-12}	14.52

Discussion. As evident from Figure 5.14a, the difference in the initial consistent points are very small, in the order of 10^{-11}. Also, DAETS and Simulink display great agreement throughout the simulation interval, as the solution differences only oscillate around the corresponding error tolerance after $t = 2$ and do not increase in value significantly. We take this as an indication that the models of DAETS and Simulink match to a high degree. As a result, our natural coordinates model is valid.

A comparison between the solution difference and global error estimates in Figure 5.14b shows a noticeable drop at the same time steps when running Simulink at its best tolerance of 3×10^{-14}, especially after $t = 1$. The unusual initial downward trend in both error types for `tol` $= 10^{-10}, 10^{-12}$ can be attributed to the deviation of about 10^{-11} between the initial consistent points of DAETS and Simulink being independent of tolerance. The global errors show a definite flattened behavior with smaller oscillation amplitudes compared to solution difference curves. According to Table 5.17, the natural coordinates model results in a high number of accurate digits considering the error tolerance used, telling of the superb level of accuracy DAETS achieves. As a whole we conclude that our natural coordinates model leads to a highly accurate solution for this problem.

5.5 Simulation Timing

Table 5.18 provides total execution times for our framework's simulation process and that of Simulink at two error tolerances for the problems presented in Chapters 5.

Table 5.18: Simulation timing data for 2D problems.

Problem	Total execution time (s)			
	`tol` $= 10^{-6}$		`tol` $= 10^{-10}$	
	DAETS	Simulink	DAETS	Simulink
N four-bar	0.03	1.17	0.04	1.21 (`ode113`)
Slide-crank	0.11	1.06	0.12	1.31 (`ode113`)
Copy machine	0.04	1.08	0.05	1.29 (`ode113`)
McPherson suspension	4.51	1.54	4.66	1.81 (`ode113`)

Chapter 6

Three-dimensional Problems

6.1 Bricard Mechanism

6.1.1 Problem Specification

Bricard mechanism is a simple multibody system comprised of thin rods and revolute joints arranged in 3D space. The description below is taken from Multibody Systems Benchmark (MBS) [17, 18]. The system has the following components:

- rods $i = 1, 2, \ldots, 5$
- ground

According to Grübler's formula [2] for calculating the number of degrees of freedom for a kinematic chain, this is an overly constrained system with zero DoF. But in practice, it can move when subjected to external forces like gravity. This is possible because of the special orientation of the revolute joints relative to each other, as depicted in Figure 6.15. More detail on computing the system DoF is provided in the subsequent subsection.

With the WF origin placed at P_0, Figure 6.15 shows how the rods are serially connected to each other using revolute joints, denoted by P_i. The first and last joints are fixed in space at P_0 and P_5, respectively, as are their joint spines; *i.e.*, the spine vectors for P_0 and P_5 are constant in the WF.

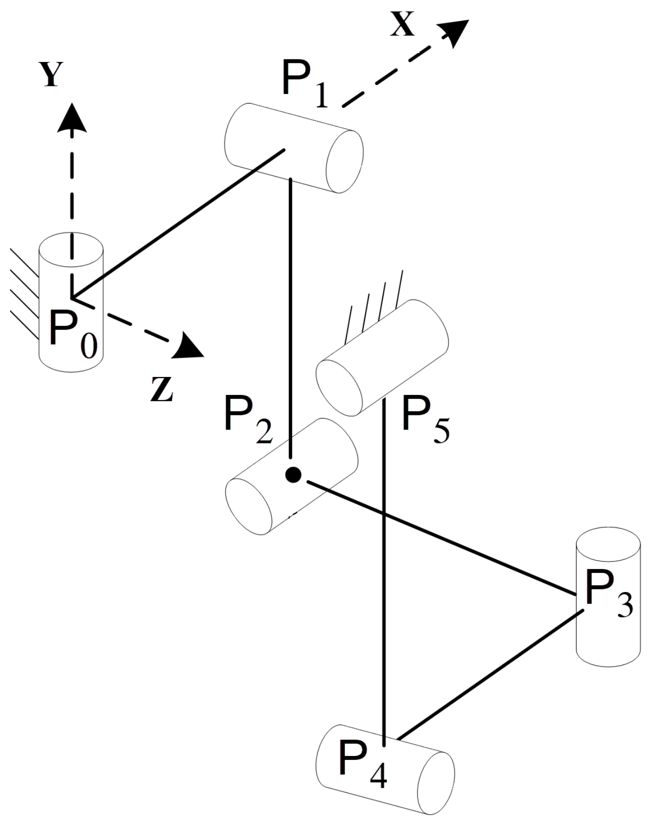


Figure 6.15: Bricard mechanism—schematic [17, 18].

The system moves solely under the effect of gravity. All rods are identical prisms, with a length of ℓ, a uniformly distributed mass of m, and a square cross-section height of h.

6.1.2 System Derivation

Coordinates. Bricard mechanism is a 3D system with five rigid bodies. Generally, this would require a generalized coordinates vector with 30 elements (five points) to uniquely define the system configuration. However, each two consecutive rods share a joint, and the first and last joints are fixed in the WF. If P_i are included in $\mathbf{q}$, the joint spine vectors can be readily calculated by the cross product of their preceding spine and rod unit vectors:

$$\begin{aligned}
\mathbf{a_0} &= (0,1,0),\\
\mathbf{a}_i &= \frac{\mathsf{P}_i - \mathsf{P}_{i-1}}{\ell} \times \mathbf{a}_{i-1}, \quad i = 1,\dots,4, \qquad (6.1.1)\\
\mathbf{a_5} &= (1,0,0),
\end{aligned}$$

where $\mathbf{a}_i$ denote the joint spine vectors. $\mathsf{P}_0 = \mathsf{O}$, and $\mathbf{a}_0$ is the joint spine at WF origin, and so on. Defining

$$\mathsf{M}_i = \frac{\mathsf{P}_i + \mathsf{P}_{i-1}}{2} \text{ and } \mathbf{u}_i = \frac{\mathsf{P}_i - \mathsf{P}_{i-1}}{\ell} \tag{6.1.2}$$

gives three unique BPVs, forming the local frame $\mathsf{M}_i\mathbf{u}_i\mathbf{a}_i$ for each rod. The $\mathbf{q}$ vector is obtained as

$$\mathbf{q} = (\mathsf{P}_{1,x}, \mathsf{P}_{1,y}, \mathsf{P}_{1,z}, \dots, \mathsf{P}_{4,x}, \mathsf{P}_{4,y}, \mathsf{P}_{4,z}).$$

Lagrangian. The kinetic and potential energy terms can be written as

$$T = \sum_{i=1}^{5} \underbrace{[\frac{1}{2}m\dot{\mathsf{M}}_i^2 + \frac{1}{2}{\omega_i}^T\mathbf{I}\omega_i]}_{\text{rod } i},$$
$$V = \sum_{i=1}^{5} \underbrace{mg\mathsf{M}_i}_{\text{rod } i},$$

where M_i is the rod midpoint defined in (6.1.2), ω_i is the angular velocity vector about each rod's local frame axes, and $\mathbf{I}$ is the mass MoI vector about the same axes, obtained from

$$\mathbf{I} = \frac{1}{12}m \begin{pmatrix} 2h^2 \\ \ell^2 + h^2 \\ \ell^2 + h^2 \end{pmatrix}.$$

Constraints. The system is constrained by two types of relations, namely the rigidity constraint of the rods, and the perpendicularity of joint spines to their following rod, which Equation (6.1.1) does not account for. Also, an additional constraint is needed to ensure that the final spine, $\mathbf{a}_5 = \mathsf{E}$, remains normal to the last rod.

$$\begin{aligned} C_{2i-2} &= \left|\mathsf{P}_i - \mathsf{P}_{i-1}\right|^2 - \ell^2, \\ C_{2i-1} &= \mathbf{a}_{i-1} \cdot \left(\mathsf{P}_i - \mathsf{P}_{i-1}\right), \end{aligned} \qquad i = 1, 2, \ldots, 5,$$

$$C_{10} = \mathbf{a}_5 \cdot \left(\mathsf{P}_5 - \mathsf{P}_4\right).$$

In total, there are 12 generalized coordinates and 11 constraints, giving the system a DoF of one, as opposed to the zero calculated by Grübler's formula [2]. Table 6.19 shows an overview of DoF calculations.

Table 6.19: Bricard mechanism—DoF calculation.

Part	Bring in	Assign	Constrain	DoF
Rod i $i = 1, 2, \ldots, 5$	P_i $(i \neq 5)$	$\mathsf{P}_0 = (0, 0, 0)$ $\mathsf{P}_5 = (0, 0, \ell)$ $\mathbf{a}_i = (\mathsf{P}_i - \mathsf{P}_{i-1}) \times \mathbf{a}_{i-1}$ $\mathbf{a}_0 = (0, 1, 0)$ $\mathbf{a}_5 = (1, 0, 0)$	$0 = \left|\mathsf{P}_i - \mathsf{P}_{i-1}\right|^2 - \ell^2$ $0 = \mathbf{a}_{i-1} \cdot (\mathsf{P}_i - \mathsf{P}_{i-1})$ $0 = \mathbf{a}_5 \cdot (\mathsf{P}_5 - \mathsf{P}_4)$	$12 - 11 = 1$
Total DoF				1

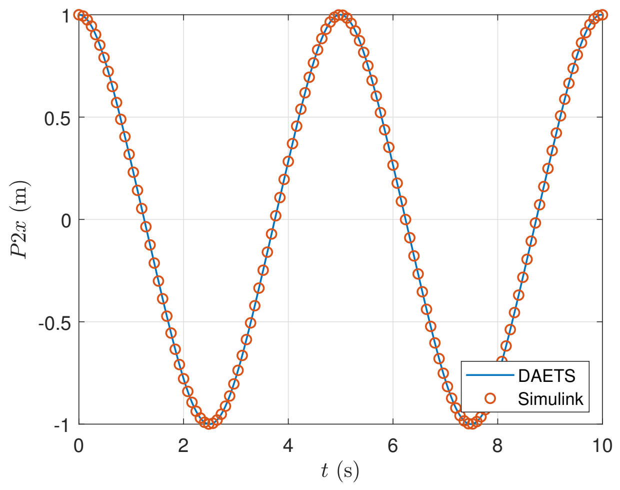
6.1.3 Numerical Results

Parameters. The results are obtained with $m = 1$ kg, $\ell = 1$ m, and $d = 0.1$ m. Joint positions and their spine vector orientations w.r.t. the WF at $t = 0$ are illustrated in Figure 6.15, while the numerical values are given in Table 6.20. All starting velocities are set to zero.

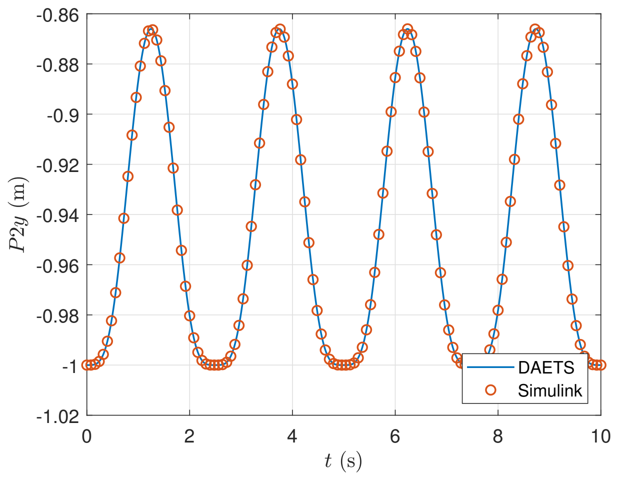
Table 6.20: Bricard mechanism—initial positions.

Joint	x	y	z	WF orientation axis
P_0	0	0	0	y
P_1	1	0	0	z
P_2	1	-1	0	x
P_3	1	-1	1	y
P_4	0	-1	1	z
P_5	0	0	1	x

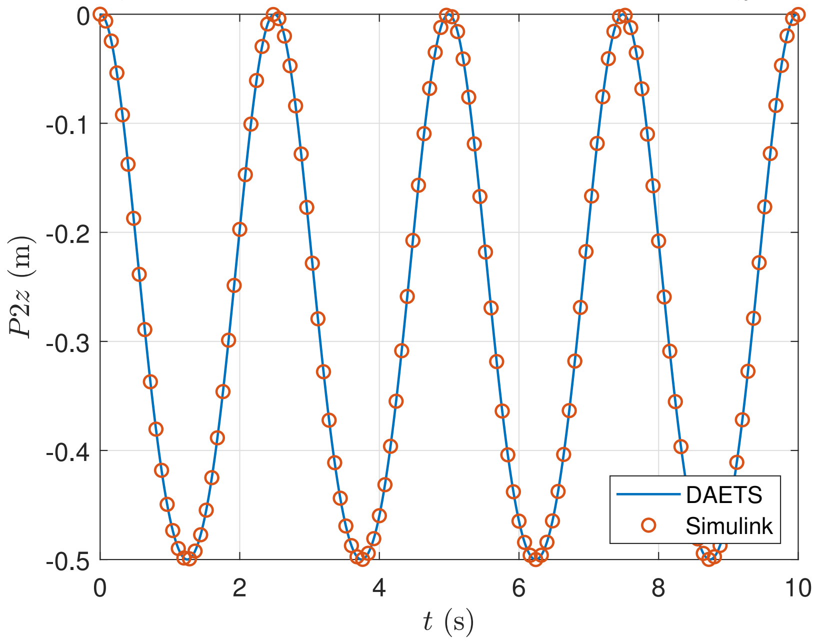
Results. Figures 6.16a, 6.16b, and 6.16c show time histories for the position coordinates of P.



(a) $\mathsf{P}_{2,x}$.



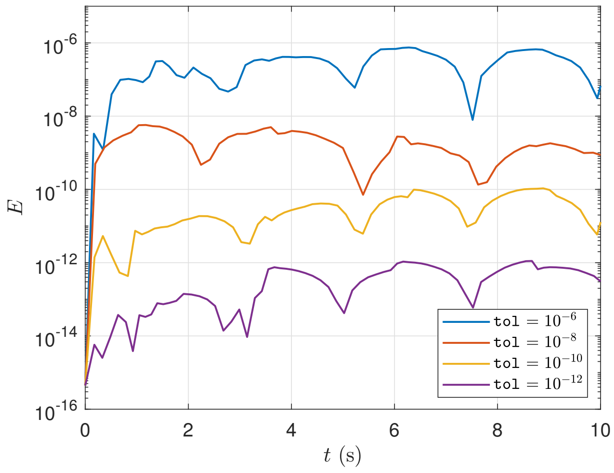
(b) $\mathsf{P}_{2,y}$.



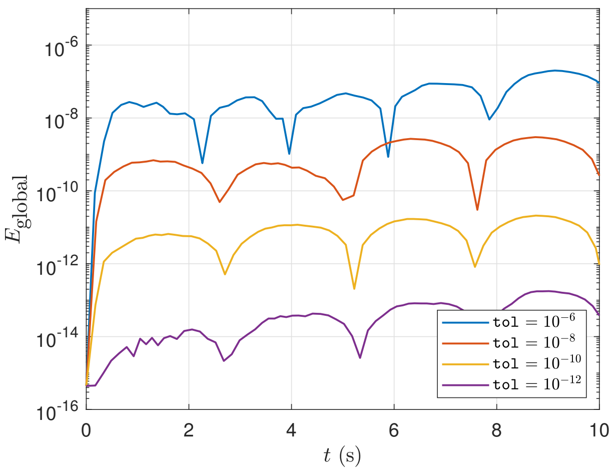
(c) $\mathsf{P}_{2,z}$.

Figure 6.16: Bricard mechanism—coordinates of P_2 versus t.

Validation. Figures 6.17a and 6.17b depict the solution difference and global error estimates between the results of Simulink and DAETS, respectively. Table 6.21 presents the minimum number of significant digits in the solution at the end of simulation.



(a) Solution difference



(b) Global error.

Figure 6.17: Bricard mechanism—difference between DAETS & Simulink solutions.

Table 6.21: Bricard mechanism—minimum number of significant correct digits in the solution at the end of simulation.

Error tolerance	SCD
10^{-6}	6.99
10^{-8}	9.52
10^{-10}	11.96
10^{-12}	13.36

Discussion. Figure 6.17 shows that both solutions start with an initial difference in the order of 10^{-15}, meaning that the calculated consistent point of DAETS is practically the same as the initial values used by Simulink. Afterwards, the solution differences in Figure 6.17a remain low, even below the respective error tolerances most of the time, indicating a high degree of agreement between DAETS and Simulink throughout the simulation period. It also shows that both models correctly represent the Bricard mechanism described in the preceding sections.

Running Simulink's model at its best tolerance of 3×10^{-14} results in global error estimates that remain entirely below the corresponding tolerances for $0 \leq t \leq 10$. Similar to solution differences, the initial value for global errors is in the order of 10^{-15}. Moreover, Table 6.21 reveals that DAETS's solutions have a high number of accurate digits when considering the error tolerance they are obtained at. As a whole, we can say that our natural coordinates model produces highly accurate results for all error tolerances.

6.2 Stiff Flyball Governor

6.2.1 Problem Specification

The flyball governor is a multibody mechanism originally devised by J. Watt in the 18th century. The model studied here is adapted from a series of benchmark problems presented in the Multibody Systems Benchmark (MBS) [17, 18]. This mechanism is similar to a classic flyball governor except that the coupler rods are replaced by spring-damper elements. The following parts constitute the system:

- shaft
- two rods
- slider
- two globes
- two spring-dampers

As shown in Figure 6.18, the world frame origin is placed at $\mathsf{W} = (0, 0, 0)$. Shaft WO is connected to the ground at the origin via a revolute joint that enables the shaft to revolve about the global z-axis. This leads to A and C moving on a circle in the global xy plane which is centered at O with a radius of $R = 0.05$ m. We denote the angle between OA and the WF x-axis by θ, and the shaft's angular velocity about the WF z-axis by

$$\omega = \frac{d\theta}{dt}.$$

Side rods AB and CD are linked to the shaft by revolute joints at A and C. The prism-shaped slider—with its center of mass denoted by E—is joined on the sides at F and G to rod midpoints, M_1 and M_2, respectively, by spring-dampers. The slider is also restricted to moving along the shaft by a friction-less prismatic joint. Furthermore, the two globes (point masses) are welded to rod

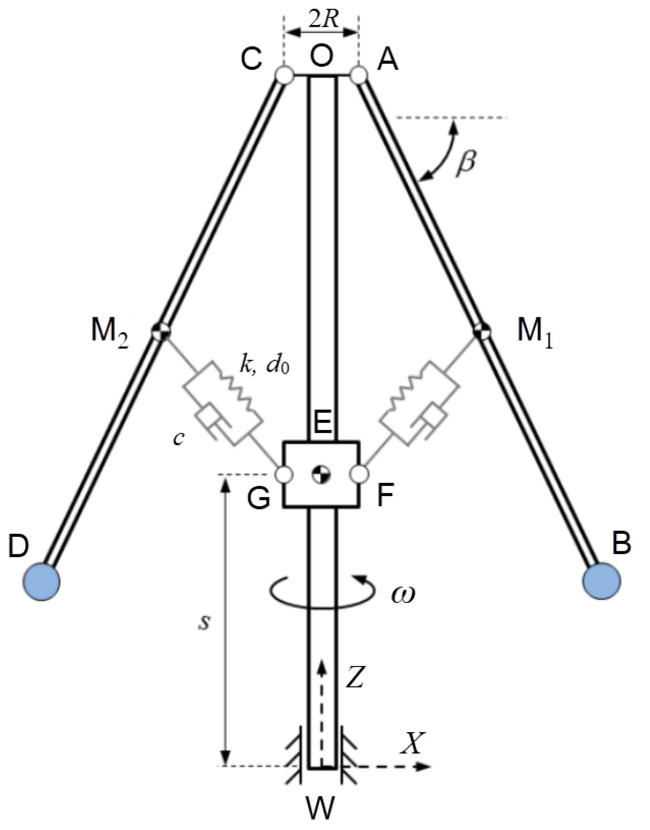


Figure 6.18: Flyball governor—schematic.

endpoints B and D. The rods and shaft have a rectangular cross-section, the dimensions of which are detailed in Table 6.22, along with other parameters.

Table 6.22: Flyball governor—system parameters.

Part	Mass	Length	Width	Height
Shaft	m_{s}	ℓ_{s}	w_{s}	h_{s}
Rods	m_{r}	ℓ_{r}	w_{r}	h_{r}
Slider	m_{sl}	ℓ_{sl}	w_{sl}	h_{sl}
Globes	m_{g}			
	Mass	Nat. length	Stiffness	Damp. coeff.
Spring-dampers	0	d_0	k	c

6.2.2 System Derivation

Coordinates. The flyball governor under study has six rigid bodies moving in 3D space. A total of six coordinates are needed to fully define their local frames and, consequently, position and orientation at any point in time, as described below.

Shaft The shaft is restricted by the revolute joint at W to only revolve about the global z-axis, meaning only a single PV is required to track its orientation, namely A. As mentioned before, θ is defined as the angle between OA and WF x-axis. With O fixed at $(0, 0, \ell_{\text{shaft}})$, A_x and A_y are added to the $\mathbf{q}$ vector. We define

$$\mathbf{u}_1 = \mathsf{O} - \mathsf{W},$$
$$\mathbf{u}_2 = \frac{\mathsf{A} - \mathsf{O}}{|\mathsf{A} - \mathsf{O}|},$$
$$\mathbf{u}_3 = \mathbf{u}_2 \times \mathbf{u}_1,$$

where $\mathbf{u}_3$ is along the spine for the revolute joints at A and C. The local frame for the shaft is formed as $\mathsf{WO}\mathbf{u}_2$.

Rods This mechanism is axis-symmetric w.r.t. the WF z-axis, eliminating the need to track both rods as they, and, consequently, the spring-dampers, move in the same manner:

$$\mathsf{C} = (-\mathsf{A}_x, -\mathsf{A}_y, \mathsf{A}_z),$$
$$\mathsf{D} = (-\mathsf{B}_x, -\mathsf{B}_y, \mathsf{B}_z).$$

Hence, we track only rod AB via the generalized coordinates vector. Point B is appended to $\mathbf{q}$. The other required PV to fully define the rod is the spine vector $\mathbf{u}_3$ which is already calculated, resulting in the local frame $\mathsf{A}\mathsf{B}\mathbf{u}_3$. Rod midpoints are also given by

$$\mathsf{M}_1 = \frac{1}{2}(\mathsf{A} + \mathsf{B}),$$
$$\mathsf{M}_2 = \frac{1}{2}(\mathsf{C} + \mathsf{D}).$$

Slider This body can only translate in the vertical direction. As such, a single component of its CoM, E_z, suffices to track its position. Since the slider body revolves with the same angular velocity about the global z-axis as the shaft, its rotation angle is also θ. The following useful points are defined on the sides of the slider:

$$\mathsf{F} = \mathsf{E} + R\mathbf{u}_2, \quad \mathsf{G} = \mathsf{E} - R\mathbf{u}_2.$$

Globes These two bodies are welded to the rods at B and D. As point masses, they need no local frames, and will not add any entries to $\mathbf{q}$.

Considering the above, the following BPVs form the generalized coordinates vector:

$$\mathbf{q} = (\mathsf{A}_x, \mathsf{A}_y, \mathsf{B}_x, \mathsf{B}_y, \mathsf{B}_z, \mathsf{E}_z).$$

Lagrangian. Denoting

J_s = shaft MoI about the global z-axis,

$\mathbf{J}_r$ = rod MoI vector about its principal axes,

ω_r = rod angular velocity vector resolved about its principal axes,

d_0 = spring natural length,

the kinetic and potential energy terms are given by

$$T = \underbrace{\frac{1}{2}J_{\mathrm{s}}\omega^2}_{\text{shaft}} + \underbrace{2 \times \frac{1}{2}\left(m_{\mathrm{r}}\dot{\mathsf{M}}_1^2 + \omega_{\mathrm{r}}\mathbf{J}_{\mathrm{r}}\omega_{\mathrm{r}}\right)}_{\text{rods}} + \underbrace{\frac{1}{2}m_{\mathrm{sl}}\dot{\mathsf{E}}^2}_{\text{slider}} + \underbrace{2 \times \frac{1}{2}m_{\mathrm{g}}\dot{\mathsf{B}}^2}_{\text{globes}},$$

$$V = g\big(\underbrace{2 \times m_{\mathrm{r}}\mathsf{M}_{2_z}}_{\text{rods}} + \underbrace{m_{\mathrm{sl}}\mathsf{E}_z}_{\text{slider}} + \underbrace{2 \times m_{\mathrm{g}}\mathsf{B}_z}_{\text{globes}}\big) + \underbrace{2 \times \frac{1}{2}k(|\mathsf{M}_1 - \mathsf{F}| - d_0)^2}_{\text{spring−dampers}}.$$

Dissipation. The Rayleigh function for modeling the dampers is given by

$$\mathcal{R} = 2 \times \frac{1}{2}c\left|(\dot{\mathsf{M}}_1 - \dot{\mathsf{F}}) \cdot \mathbf{u}_4\right|^2,$$

where $\mathbf{u}_4$ is a unit vector parallel to the spring-damper $\mathsf{M}_1\mathsf{F}$:

$$\mathbf{u}_4 = \frac{\mathsf{M}_1 - \mathsf{F}}{|\mathsf{M}_1 - \mathsf{F}|}.$$

Constraints. The constraint introduced to the system by the prismatic joint connecting the slider to the shaft is accounted for implicitly by setting the global x and y positions of E to zero. Hence, the only constraints that need to be enforced explicitly are the constancy of the distance from A to O, the constant length of rod AB, and the perpendicularity of AB to the joint spine $\mathbf{u}_3$. The second rod does not require any constraints as it is not represented in $\mathbf{q}$; therefore

$$C_1 = |\mathsf{A} - \mathsf{O}|^2 - R^2,$$

$$C_2 = |\mathsf{A} - \mathsf{B}|^2 - \ell_{\mathrm{s}}^2,$$

$$C_3 = (\mathsf{A} - \mathsf{B}) \cdot \mathbf{u}_3.$$

A summary of DoF calculations is presented in Table 6.23. Although the table computes a total DoF equal to 3, it is worth noting that the DoF of the actual system is 4. This is caused by our using the symmetry in the mechanism to eliminate rod CD from the generalized coordinates vector, which leads to a reduced system size. Instead of tracking the rod directly using $\mathbf{q}$, we find its position and velocity via mirroring those of rod AB w.r.t. the WF z-axis.

Table 6.23: Flyball governor—DoF calculation.

Part	Bring in	Assign	Constrain	DoF
Shaft	A_x A_y	$\mathsf{O} = (0, 0, \ell_\mathrm{s})$ $\mathbf{u}_1 = \mathsf{O}$ $\mathbf{u}_2 = \frac{\mathsf{A}-\mathsf{O}}{\lvert\mathsf{A}-\mathsf{O}\rvert}$ $\mathbf{u}_3 = \mathbf{u}_2 \times \mathbf{u}_1$	$0 = \lvert\mathsf{A} - \mathsf{O}\rvert^2 - R^2$	$2 - 1 = 1$
Rod AB	B_x B_y B_z	$\mathsf{M}_1 = \frac{1}{2}(\mathsf{A} + \mathsf{B})$	$0 = \lvert\mathsf{A} - \mathsf{B}\rvert^2 - \ell_\mathrm{s}^2$ $0 = (\mathsf{A} - \mathsf{B}) \cdot \mathbf{u}_3$	$3 - 2 = 1$
Slider	E_z	$\mathsf{F} = \mathsf{E} + R\mathbf{u}_2$ $\mathsf{G} = \mathsf{E} - R\mathbf{u}_2$		$1 - 0 = 1$
Globe B				
Total DoF				3

6.2.3 Numerical Results

Parameters. The numerical value of parameters used for simulation are provided in Table 6.29. At $t = 0$ s the spine of the two top revolute joints is parallel to the global y-axis. At the same time, the shaft is spun with an initial angular velocity of $\omega_0 = 2\pi$ rad/s, while $\dot{\mathsf{A}}_y$ and $\dot{\mathsf{B}}_y$ have their initial values defined as

$$\begin{aligned} \dot{\mathsf{A}}_{y,0} &= \omega_0 R, \\ \dot{\mathsf{B}}_{y,0} &= \omega_0 \big(R + \ell_\mathrm{r} \cos\beta_0\big). \end{aligned}$$

All initial values are given in Table 6.31.

Results. The graphs in Figure 6.19 depict the time history of the position coordinates of B, as well as E_z for comparison with the results of Simulink. The numerical values used for parameters lead to a stiff system, and we had to switch to DAETS's stiff solver to simulate the mechanism in a timely manner.

Table 6.24: Flyball governor—simulation parameters.

Part	Mass	Length	Width	Height
Shaft	0.3	1	0.01	0.01
Rods	0.3	1	0.01	0.01
Slider	3	0.1	0.1	0.1
Globes	5			
	Mass	Nat. length	Stiffness	Damp. coeff.
Spring-dampers	0	0.5	8×10^5	4×10^4

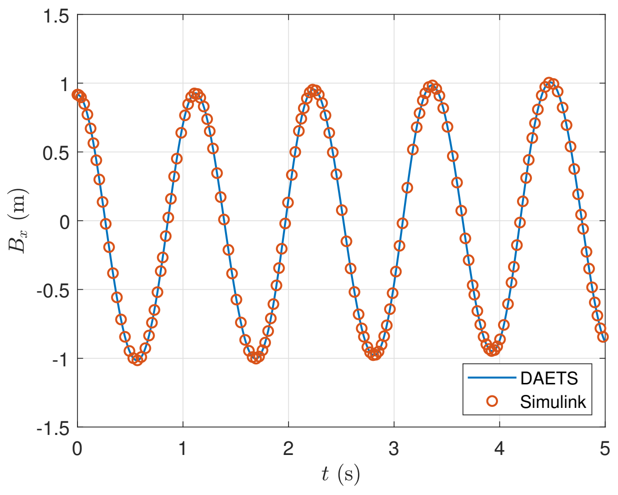
Table 6.25: Flyball governor—initial values.

Variable	x	$\dot{x}$
A_x	0.05	0
A_y	0	0.314
B_x	0.916	0
B_y	0	5.76
B_z	0.5	0
E_z	0	0

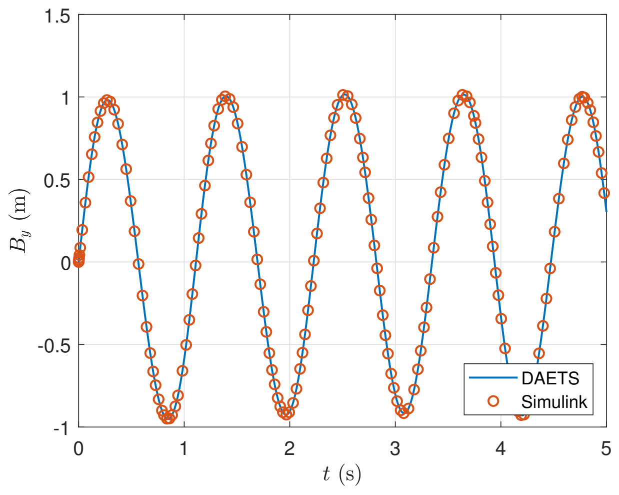
Validation. Solution differences are plotted in Figure 6.20a, while global errors are shown in Figure 6.20b. Table 6.26 provides the minimum number of significant digits in the solution at the end of simulation.

Discussion. The system has two dampers, but their line-of-sight passes through the shaft, meaning that they generate no torque about OW. As all external forces are parallel to the shaft, the total angular momentum of the system w.r.t. shaft axis is conserved. As such, system motion is expected to continue with the same periodic pattern evident in Figure 6.19 without being damped. The solution differences for all tolerances have a starting value in of order 10^{-16}, showing an excellent agreement between the initial consistent points of DAETS and Simulink. Furthermore, it is evident from Figure 6.20a that the growth rate of errors is slightly lower for stricter tolerance values. In general, a high level of agreement between the results of Simulink and DAETS is observed, proving the validity of our natural coordinates model.

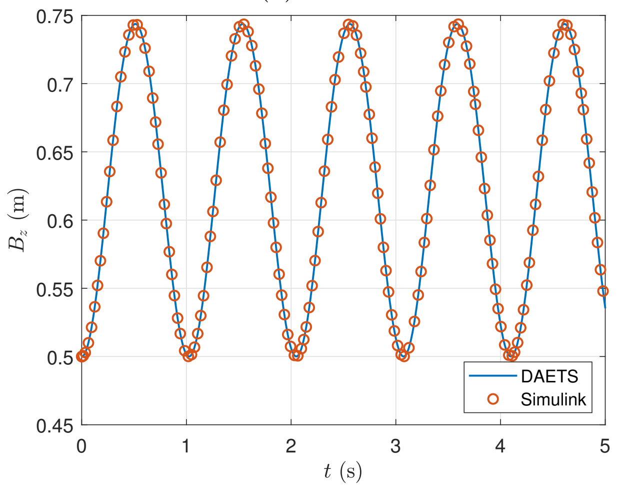
A comparison between the solution difference and global error curves does not show a significant reduction in value when setting Simulink's tolerance to 3×10^{-14}. Global errors estimates in



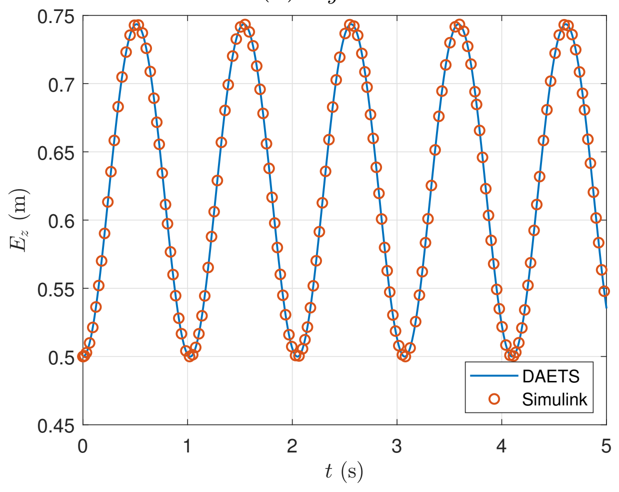
(a) B_x.



(b) B_y.



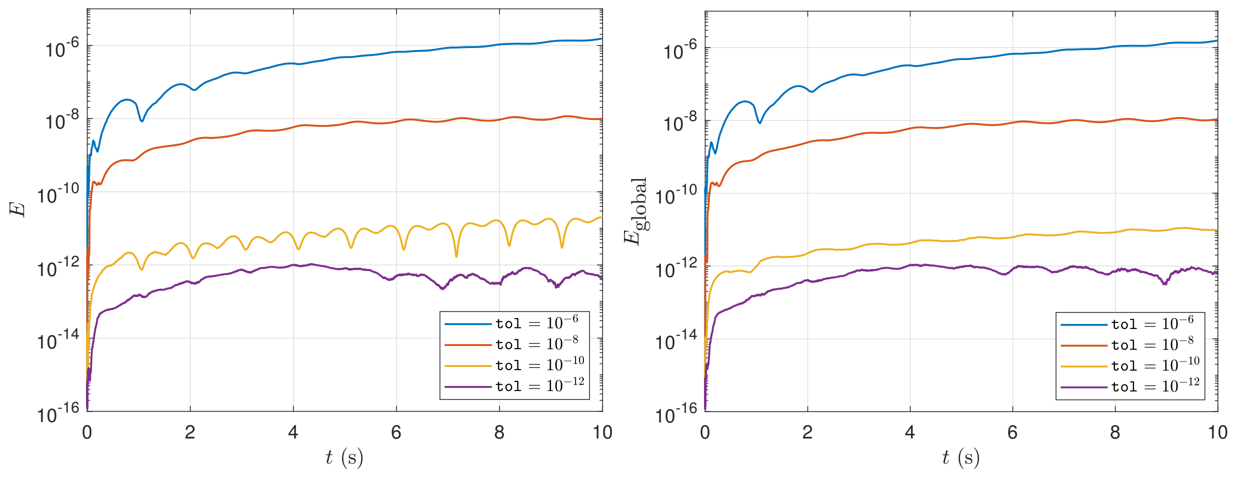
(c) B_z.



(d) E_z.

Figure 6.19: Flyball governor—coordinates of B and E_z versus t.

Figure 6.20b have a starting value in the order of 10^{-15}, demonstrating a very high degree of similarity between the initial consistent points computed by DAETS and Simulink. Per the data in Table 6.26, DAETS produces a reasonable number of accurate digits for the corresponding error tolerances. As a whole, we can say the acquired data verifies that our natural coordinates model produces an accurate solution for all error tolerances.



(a) Solution difference. (b) Global error.

Figure 6.20: Flyball governor—difference between DAETS and Simulink solutions.

Table 6.26: Flyball governor—minimum number of significant correct digits in the solution at the end of simulation.

Error tolerance	SCD
10^{-6}	5.78
10^{-8}	7.93
10^{-10}	10.99
10^{-12}	12.15

6.3 Five-link Suspension System

6.3.1 Problem Specification

The five-link suspension is a well-known subtype of multi-link suspension mechanisms used in vehicles. We study the system described in [46, Section 4.1], see Figure 6.21. Figure 6.22 displays the side and front views of this mechanism in its initial state.

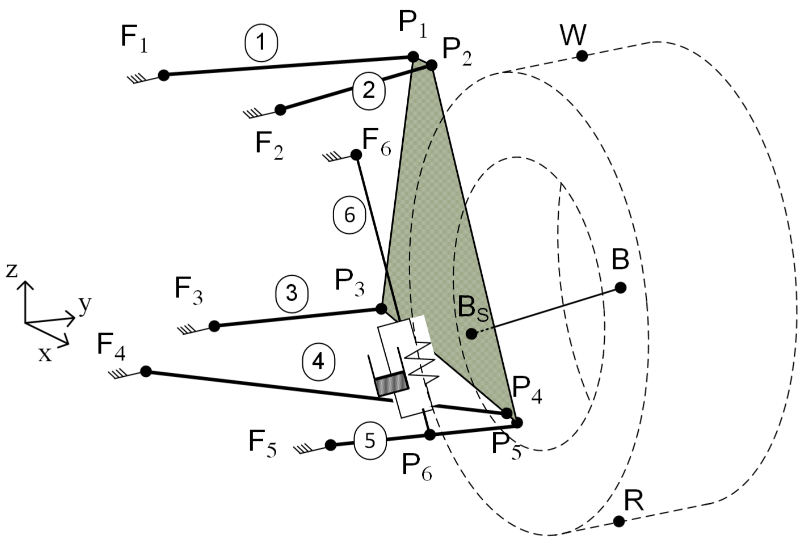
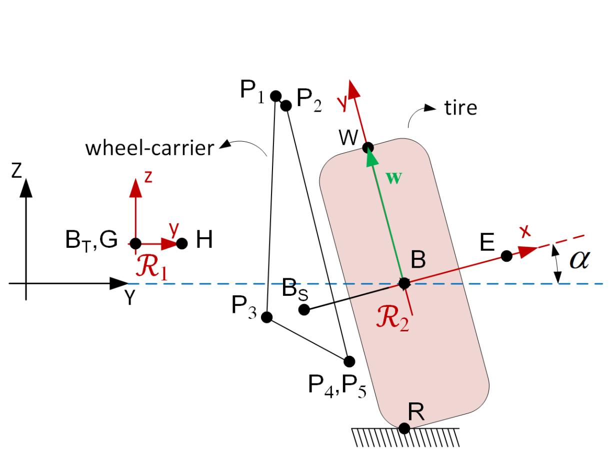


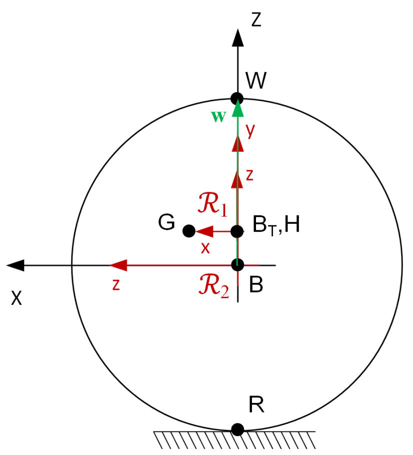
Figure 6.21: Five-link suspension—isometric view.

This system consists of a car body, five rods connecting it to a wheel-carrier plate, and a tire. A spherical joint connects rod $i = 1, \ldots, 5$, to the car body at fixed points point F_i and to the plate at points P_i. A massless spring-damper (called "primary") joins P_6, the midpoint of rod 5, to the car body at F_6. The wheel axis $\mathsf{BB_S}$ is rigidly connected to the plate. A revolute joint connects the wheel-carrier and the tire at B, allowing for relative rotation about $\mathsf{BB_S}$.

The car body—which acts as the inertial frame—moves with a linear velocity v in the global x direction. Assuming no slip conditions, the tire revolves with constant angular velocity ω about the wheel axis. If friction effects are ignored everywhere but for the road, the external forces acting on the system stem from gravity, the primary spring-damper, and the interaction of the road and tire. This interaction is modeled using a linear spring (the "tire spring") connecting the wheel's center of mass B to point R , which denotes the road-tire contact point. The tire is assumed to always



(a) side view



(b) front view

Figure 6.22: Five-link suspension—side and front views.

remain in contact with the road.

Road shape determines the movement of R and, consequently, the force exerted on the suspension via the tire spring. A sinusoidal profile is implemented to represent a rough road via

$$\mathsf{R}_x = \mathsf{B}_{x0},$$

$$\mathsf{R}_y = \mathsf{B}_{y0},$$

$$\mathsf{R}_z = R_{amp}\sin(\frac{2\pi}{R_{per}}t),$$

where the $_0$ subscript denotes initial values.

All rods are cylindrical in shape. For rod i, we denote length ℓ_i, radius r, mass m_i and MoI J_i relative to F_i and an arbitrary axis perpendicular to the vector $\mathsf{P}_i - \mathsf{F}_i$. By R_0 we denote the radius of the tire. The parameter notation for the spring elements is presented in Table 6.27.

Table 6.27: Five-link suspension—spring elements notation.

	Mass	Nat. length	Stiffness	Damp. coeff.
Primary spring-damper	0	d_p	k_p	c_p
Tire spring	0	d_T	k_T	0

The wheel-carrier has a mass of m_wc, with the MoI w.r.t. its CoM B_T and the inertial reference frame denoted by $\mathbf{J}_{\text{wc}}$. The tire's mass is m_{T} and its MoI tensor, relative to point B and the tire's principal axes (with the local x-axis being $\mathsf{B}_{\text{S}}\mathsf{B}$), is denoted by $\mathbf{J}_{\text{T}}$. Finally, the camber angle is denoted by α in Figure 6.22a.

6.3.2 System Derivation

Coordinates. There are seven rigid bodies: five rods, the wheel-carrier, and the tire. Each has six degrees of freedom; however, one can reduce the total number of components in the generalized coordinates vector $\mathbf{q}$ by considering the geometric relations between rigid bodies and the constraints they impose, leading to the following local frames and PVs:

Wheel-carrier. To create a local frame, whose axes are coincident with those of the WF at $t = 0$, the virtual points G and H are added to this body. Their initial position in the WF are defined by

$$\mathsf{G}_0 = \mathsf{B}_{\text{T0}} + (\ell_x, 0, 0) \quad \text{and} \quad \mathsf{H}_0 = \mathsf{B}_{\text{T0}} + (0, \ell_y, 0)\,.$$

With B_{T} acting as the origin, we form the local frame $\mathsf{B}_{\text{T}}\mathsf{GH}$, denoted by $\mathcal{R}_1$ in Figure 6.22. We chose the coordinates of B_{T}, G and H as members of the $\mathbf{q}$ vector. Denote by $\mathbf{Q}$ the associated rotation matrix. The positions and velocities of P_i, B, B_{S}, and E are computed as shown in Table 6.28.

Tire. The revolute joint at B restricts the motion of this body to rotation-only (relative to the wheel-carrier) about $\mathsf{B}_{\text{S}}\mathsf{B}$. The forces acting on the tire are gravity, the reaction of the revolute joint, and the road reaction force. The road force is assumed to keep the tire rotating at constant angular velocity. One way to simulate this is to disconnect the tire from the road and, instead, give it an initial angular velocity of ω about the wheel axis. Then, only the first two forces are left, which generate no torque about $\mathsf{B}_{\text{S}}\mathsf{B}$ because their vectors pass through it. As a result, there is no external torque acting on the tire and the magnitude of its angular momentum about $\mathsf{B_SB}$ is conserved. Thus, its angular velocity w.r.t. the tire axis remains unchanged.

W denotes a point on the surface of the tire and its initial position is shown in Figure 6.22. It lies on a line originating from B that is always perpendicular to $\mathsf{B_SB}$. Adding W to $\mathbf{q}$ helps us keep track of the tire's orientation in space. With B and E already at hand from the wheel-carrier, the body-fixed frame BEW is established. It has its origin at B, and the local x and y axes pointing toward E and W, respectively. The frame is denoted by $\mathcal{R}_2$ in Figure 6.22.

Rods. The endpoint F_i on each rod is fixed to the car body via a spherical joint. The other endpoint, namely P_i, is already computed from the wheel-carrier frame. Given the cylindrical shape of the rods they are rotationally symmetric w.r.t. their primary axis, defined as the vector $\mathsf{P}_i - \mathsf{F}_i$. As a result, no more PVs are needed to fully calculate the position and energy terms of the rods and no local frame needs to be formed.

Finally, the generalized coordinates vector with 12 components is obtained as

$$\mathbf{q} = (\mathsf{G}, \mathsf{H}, \mathsf{B}_{\mathrm{T}}, \mathsf{W}).$$

Lagrangian. The kinetic and potential energies are given by

$$T = \underbrace{\frac{1}{2}(m_{\mathrm{wc}}\dot{\mathsf{B}}^2_{\mathrm{tire}} + \omega^T_{\mathrm{wc}}\mathbf{J}_{\mathrm{wc}}\omega_{\mathrm{wc}})}_{\mathrm{wheel-carrier}} + \underbrace{\frac{1}{2}(m_{\mathrm{T}}\,\dot{\mathsf{B}}^2 + \omega^T_{\mathrm{T}}\mathbf{J}_{\mathrm{T}}\omega_{\mathrm{T}})}_{\mathrm{tire}} + \underbrace{\sum_{i=1}^{5}\frac{1}{2}\omega_i^T\mathbf{J}_i\omega_i}_{\mathrm{rods}}\,, \tag{6.3.1}$$

$$V = \underbrace{\frac{1}{2}k_p|\mathsf{P}_6 - \mathsf{F}_6|^2}_{\mathrm{primary\ spring}} + \underbrace{\frac{1}{2}k_{\mathrm{T}}|\mathsf{B} - \mathsf{U}|^2}_{\mathrm{tire\ spring}} + \underbrace{m_{\mathrm{wc}}g\mathsf{B}_{\mathrm{T}z}}_{\mathrm{wheel-carrier}} + \underbrace{m_{\mathrm{T}}g\mathsf{B}_z}_{\mathrm{tire}} + \underbrace{\sum_{i=1}^{5}m_ig\left(\frac{\mathsf{P}_{i,z} + \mathsf{F}_{i,z}}{2}\right)}_{\mathrm{rods}}, \tag{6.3.2}$$

where the $_{\mathrm{wc}}$ and $_{\mathrm{T}}$ subscripts refer to the wheel-carrier and tire, respectively.

Wheel-carrier and tire angular velocity vectors are computed relative to the body-fixed local frames defined above. Those of the rods, however, are calculated w.r.t. the frame $\mathsf{F}_i\mathsf{P}_i\mathbf{s}_i$, where $\mathbf{s}_i$

is a unit vector passing through F_i that is perpendicular to each rod's primary axis, $\mathsf{P}_i - \mathsf{F}_i$.

$$\mathbf{s}_i = \frac{\mathsf{P}_i - \mathsf{F}_i}{\left|\mathsf{P}_i - \mathsf{F}_i\right|} \times \frac{\dot{\mathsf{P}}_i}{\left|\dot{\mathsf{P}}_i\right|}.$$

As seen in (6.3.1), the translational kinetic energy term for the rods is eliminated. The logic behind this is that the spherical joints at F_i enable us to view the movement of the rods as pure rotation (per Euler's Theorem [16, Chapter 4]) about $\mathbf{s}_i$ with the angular velocity

$$\omega_i = \big(0, |\dot{\mathsf{P}}_i|/\ell_i, 0\big).$$

Dissipation. We model the effect of the primary damper using Rayleigh's dissipation function

$$\mathcal{R} = \frac{1}{2}c_p\left(\frac{\dot{\mathsf{P}}_6 \cdot (\mathsf{P}_6 - \mathsf{F}_6)}{|\mathsf{P}_6 - \mathsf{F}_6|}\right)^2.$$

Constraints. There are three rigidity constraints for the wheel-carrier, one for the rigidity of the tire and another for its restricted motion, and additional length constraint brought on by each rod.

$$\begin{aligned}
C_1 &= (\mathsf{B}_\mathrm{T} - \mathsf{G})^2 - \ell_x^2,\\
C_2 &= (\mathsf{B}_\mathrm{T} - \mathsf{H})^2 - \ell_y^2,\\
C_3 &= (\mathsf{B}_\mathrm{T} - \mathsf{G}) \cdot (\mathsf{B}_\mathrm{T} - \mathsf{H}),\\
C_4 &= (\mathsf{W} - \mathsf{B}) - {R_0}^2,\\
C_5 &= (\mathsf{W} - \mathsf{B}) \cdot (\mathsf{B} - \mathsf{B}_\mathrm{S}),\\
C_{i+5} &= (\mathsf{P}_i - \mathsf{F}_i)^2 - {\ell_i}^2; \quad i = 1, \ldots, 5\,.
\end{aligned}$$

The original natural coordinates model developed by von Schwerin [46] for the five-link suspension mechanism has 45 generalized coordinates and 43 constraint equations. They use simplifying assumptions to transform their system into one with 25 mixed coordinates (*i.e.*, angles are added to the model as variables) and 23 constraints. For comparison, our model only requires 12 generalized coordinates and 10 kinematic constraints.

Table 6.28: Five-link suspension—DoF calculations.

Part	Bring in	Assign	Constrain	DoF
wheel-carrier	G H B_T	$\mathsf{G}_0 = \mathsf{B}_{\mathrm{T}0} + (\ell_x, 0, 0)$ $\mathsf{H}_0 = \mathsf{B}_{\mathrm{T}0} + (0, \ell_y, 0)\,.$ $\mathsf{P}_i = \mathsf{B}_\mathrm{T} + \mathbf{Q}(\mathsf{P}_{i_0} - \mathsf{B}_{\mathrm{T}_0})$ $\mathsf{B} = \mathsf{B}_\mathrm{T} + \mathbf{Q}(\mathsf{B}_0 - \mathsf{B}_{\mathrm{T}_0})$ $\mathsf{B}_\mathrm{S} = \mathsf{B}_\mathrm{T} + \mathbf{Q}(\mathsf{B}_{\mathrm{S}0} - \mathsf{B}_{\mathrm{T}_0})$ $\mathsf{E} = 2\mathsf{B} - \mathsf{B}_\mathrm{S}$	$0 = (\mathsf{B}_\mathrm{T} - \mathsf{G})^2 - \ell_x^2$ $0 = (\mathsf{B}_\mathrm{T} - \mathsf{H})^2 - \ell_y^2$ $0 = (\mathsf{B}_\mathrm{T} - \mathsf{G}) \cdot (\mathsf{B}_\mathrm{T} - \mathsf{H})$	$9 - 3 = 6$
rod i $i = 1, \ldots, 5$			$0 = (\mathsf{P}_i - \mathsf{F}_i)^2 - \ell_i{}^2$	$0 - 5 = -5$
tire	W		$0 = (\mathsf{W} - \mathsf{B}) - R_0{}^2$ $0 = (\mathsf{W} - \mathsf{B}) \cdot (\mathsf{B} - \mathsf{B}_\mathrm{S})$	$3 - 2 = 1$
Total DoF				2

6.3.3 Numerical Results

Parameters. All the rods have the same radius of 0.02 m and R_0, the tire radius, is 0.3 m. Road profile parameters are $R_{amp} = 0.03$ m and $R_{per} = 0.2$ s. Numerical values for the other parameters are given in Table 6.29. The global Cartesian coordinates of fixed points are given in Table 6.30, and Table 6.31 presents initial values w.r.t. WF, both in millimeters. All starting velocities are zero except for

$$\mathsf{W}_{x,0} = R_0\omega,$$

where $\omega = 100$ rad/s.

Results. Figure 6.23 illustrates the time history of the global Cartesian position coordinates of B_T, as well as the suspension camber angle in degrees.

Validation. We present the 2-norm of the solution differences in Figure 6.24a, and the global errors in Figure 6.24b. Table 6.32 provides the minimum number of significant digits in the solution at the end of simulation.

Table 6.29: Five-link mechanism—simulation parameters.

Part	Mass	Length	MoI
Wheel-carrier	15		$\begin{pmatrix} 0.3 & 0.01 & 0.015 \\ 0.01 & 0.2 & 0.05 \\ 0.015 & 0.05 & 0.2 \end{pmatrix}$
Tire	12		$\begin{pmatrix} 1.46 & 0 & 0 \\ 0 & 0.74 & 0 \\ 0 & 0 & 0.74 \end{pmatrix}$
Rod 1	2	0.22	0.03
Rod 2	2	0.28	0.05
Rod 3	4	0.30	0.12
Rod 4	1	0.42	0.06
Rod 5	3	0.41	0.17

	Mass	Nat. length	Stiffness	Damp. coeff.
Primary spring-damper	0	0.88	1.4×10^5	1400
Tire spring	0	0.30	2×10^6	0

Table 6.30: Five-link mechanism—fixed positions.

Point	x	y	z
F_1	-64.0	413.0	327.0
F_2	-303.0	432.0	295.0
F_3	-93.0	366.0	4.0
F_4	-236.0	388.0	-109.0
F_5	211.5	384.5	-100.0

Discussion. Both the solution difference and global error plots start from a value in the order of 10^{-16}, which is telling of an excellent degree of similarity in the initial consistent points determined by Simulink and DAETS. All relative difference plots display a positive trend with roughly the same growth rate for all tolerances, as we can see all differences have increased by a factor of about 10^5 by the end of the simulation. As a result, we can say that DAETS and Simulink both show the same kinematic and dynamical properties which confirms the validity of our natural coordinates model.

The global error estimates in Figure 6.24b show a similar behavior as the solution differences. There is no significant reduction in value when comparing global errors to solution differences at the same time steps. However, the errors are reasonable for each tolerance. We can also see a

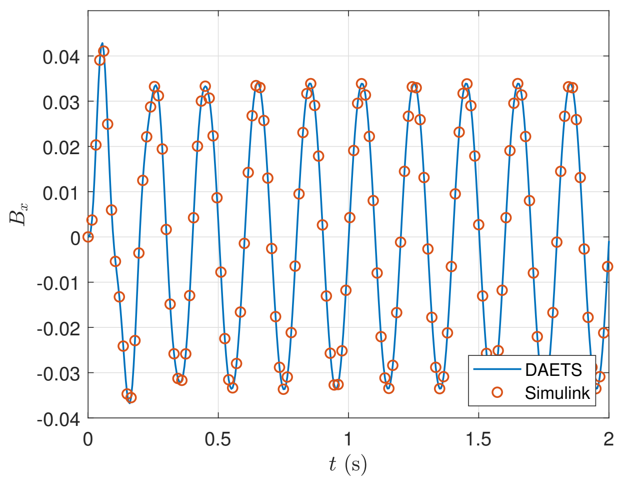
Table 6.31: Five-link mechanism—initial values.

Point	x	y	z	$\dot{x}$	$\dot{y}$	$\dot{z}$
G	-64.0	413.0	327.0	0	0	0
H	-303.0	432.0	295.0	0	0	0
B_T	-103.0	463.0	87.0	0	0	0
W	-93.0	366.0	4.0	3×10^4	0	0
B	0	768.0	0	0	0	0
B_S	0	686.0	-1.0	0	0	0
P_1	-64.0	636.0	345.0	0	0	0
P_2	-117.0	636.0	338.0	0	0	0
P_3	-188.0	647.0	-23.0	0	0	0
P_4	-5.0	737.0	-130.0	0	0	0
P_5	2.5	737.0	-134.0	0	0	0

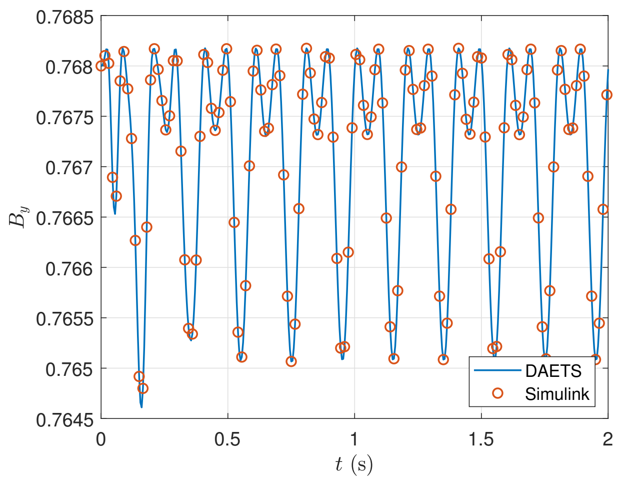
Table 6.32: Five-link suspension—minimum number of significant correct digits in the solution at the end of simulation.

Error tolerance	SCD
10^{-6}	1.60
10^{-8}	3.56
10^{-10}	5.68
10^{-12}	7.99

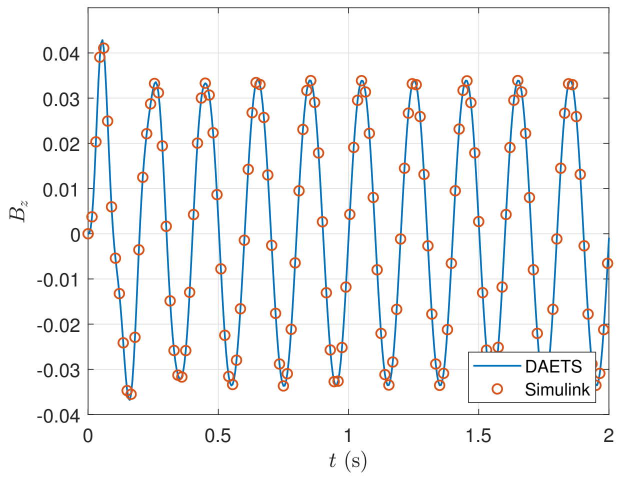
proportionality to tolerance values since decreasing the tolerance by a factor of 100 results in the global errors being reduced by, at least, the same factor. Given the above, we can conclude that DAETS achieves a good level of accuracy in this problem.



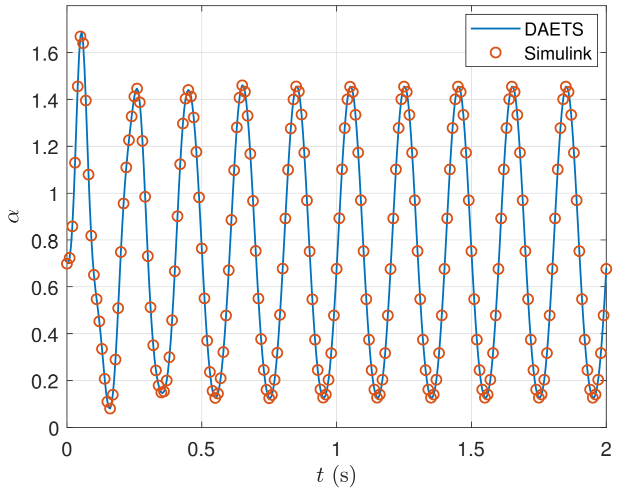
(a) B_x.



(b) B_y.

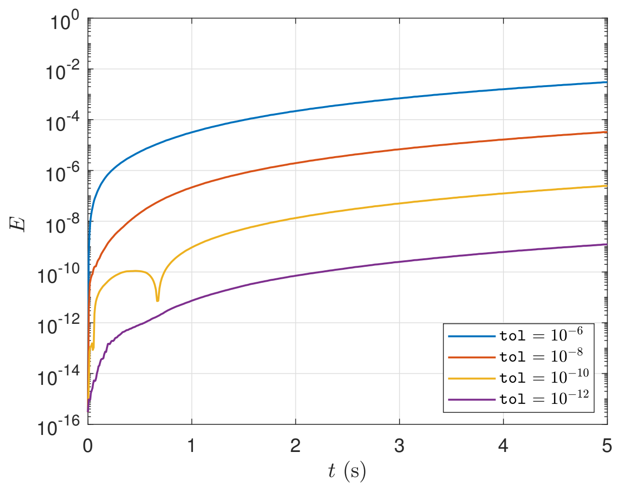


(c) B_z.

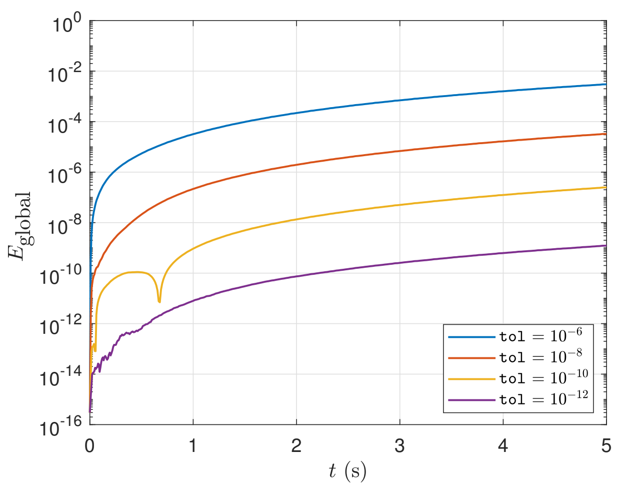


(d) Camber angle.

Figure 6.23: Five-link suspension—position coordinates of B and the camber angle versus t.



(a) Solution difference.



(b) Global error.

Figure 6.24: Five-link suspension—difference between DAETS and Simulink solutions.

6.4 Simulation Timing

Table 6.33 provides total execution times for our framework's simulation process and that of Simulink at two error tolerances for the problems presented in this chapter. On the flyball governor problem, we used the stiff solver in DAETS. In the future, we will investigate how to decrease the running time of this solver.

Table 6.33: Simulation timing data for 3D problems.

Problem	Total execution time (s)			
	`tol` $= 10^{-6}$		`tol` $= 10^{-10}$	
	DAETS	Simulink	DAETS	Simulink
Bricard mechanism	0.25	1.03	0.32	1.05 (`ode113`)
Flyball governor	434.97	1.58	709.79	2.11 (`ode15s`)
Five-link suspension	3.19	2.61	2.95	3.32 (`ode113`)

Chapter 7

Conclusion

7.1 Summary

This thesis presents a framework for the modeling and simulation of multibody systems based on the C++ computational package DAETS. We utilize a natural coordinates method that offers an intuitive procedure for developing models to capture the kinematic and dynamical behavior of planar and spatial mechanisms. A key point is that users do not need to derive the governing equations. Instead, they provide a Lagrangian term, kinematic constraints, as well as Rayleigh's dissipation function and external forces/torques if applicable. The equations are then generated via automatic differentiation. The resulting model is solved by a Taylor series numerical integrator capable of dealing with high-index DAEs and non-constant mass matrices efficiently. Our natural coordinates model requires only three PVs to form local frames for a rigid body, as opposed to the currently prevailing techniques that need four PVs. We also do not use angles as generalized coordinates, avoiding the associated trigonometric functions that inevitably appear. This gives us a decided advantage over the existing multibody analysis software as our models tend to be simple and easy-to-understand.

The chief focus of this thesis is developing natural coordinates models for multibody systems and simulating them. Based on the results of case studies on seven different mechanisms, we argue that our framework can perform remarkably accurate simulations for multibody systems with varying complexities. To achieve this, we validate and verify the models using reference solutions obtained from multibody models developed in Simulink. The structure of each problem in Chapters 2 and 3 reflects the steps integral to our process of studying multibody systems: specifying the physical properties of a mechanism, transforming it into a mathematical model, and solving the model with the numerical integrator of DAETS to obtain the desired output. We believe this approach offers a systematic and intuitive procedure for analyzing mechanisms. Simulink is used as the source of references solutions due to a host of useful features, including rapid model creation and ease of access to position/velocity data for any point of the mechanism. We present our results in the form of position coordinate time histories for direct visual comparison, as well as two types of relative error. Information on the accuracy of the initial consistent point of DAETS, the accumulation of numerical errors, as well as the trend of errors over time can all be gleaned from this data.

An important area in which this framework could be of use is teaching. This is especially true in the case of undergraduate students learning the basics of multibody modeling in courses like machine dynamics or mechanical vibrations. When the goal is to delve into the mathematical model that underpins Lagrangian mechanics, encoding a mechanism using our natural coordinates model can be instructive, as it requires the students to write down the energy terms as well as the constraints. This allows them to understand how each term is constructed from system coordinates, and also see the contribution of each element to the Lagrangian. Moreover, deriving the constraint equations grants insight into the geometric relations inherent to the mechanism. Using natural coordinates, the students will intuitively learn how to take advantage of such relations to eliminate unknown variables and reduce model size. Furthermore, compared to the Newtonian approach, our natural coordinates model enables one to better understand the effect of each joint type on the motion of mechanism bodies, because this effect directly manifests itself in the constraints in terms of generalized coordinates.

On the other hand, if the goal is experimenting with different mechanisms, our framework can accommodate this need with its MechSpec facility. MechSpec takes high-level descriptions of mechanisms that are easy to develop and produces solution data, as well as an animation of the mechanism. This tool can be utilized to perform quick and easy-to-setup simulations for a variety of planar mechanisms, which we believe to be useful in a teaching environment, especially those with time constraints. In the experience of the author, MechSpec is also much faster to learn and work with compared to existing multibody simulation suites, such as ADAMS.

7.2 Future Works

DAETS allows for simulating an impressive variety of mechanisms. However, there are limitations as to the extent of physical phenomena it can incorporate into a model, such as some types of nonholonomic constraints, and external forces/torques dependent on position. Currently, some of these elements could potentially be modeled by DAETS through special "tricks", like the approach we used for the five-link mechanism that effectively removed the road-tire rolling contact, a nonholonomic element. Still, this thesis largely does not deal with such systems. Considering the above, as well as with the experience gained over the course of this research project, we put forth the following recommendations for future works.

With regard to the development of DAETS, there are a number of missing features that, if added, would greatly expand the range of systems that can be modeled. As mentioned before, one such feature is support for nonholonomic constraints, such as those imposed by constant velocity joints. Another example is externally applied forces/torques that depend on variables other than time, which are essential for modeling special spring-dampers and actuators. Lastly, adding support for discontinuous functions would grant DAETS the ability to model sharp road bumps and depressions, impact events, joint dislocations, etc., without resorting to approximation methods. Another potentially useful addition to DAETS is an inverse dynamics toolset, needed for determining motor/actuator output or joint reactions required for a given motion path. The natural next step would then be adding parameterization capabilities which, coupled with an optimization module, could turn our system into a fully-fledged multibody design engine.

Future works could also be dedicated to the study of more complex multibody systems that result in DAEs with a high number of variables, with a focus on assessing computational performance. Profiling the simulation process of sufficiently large systems can grant a realistic view of the run times required by different parts of the analysis process. This data can be utilized to optimize the code and increase numerical efficiency in general. Moreover, combined with the validation results obtained in this thesis, computational cost data can form a complete viability study for DAETS for the purpose of comparison with existing multibody simulation tools.

Appendix A

DAETS Code Sample: The Copy Machine Problem

The codes displayed in Listings A.1, A.2, and A.3 put together constitute the C++ code used to simulate the copy machine mechanism described in Section 5.3. Listing A.1 shows the code preamble. Here we define the model parameters `nq` and `nc`, which hold the length of the generalized coordinates vector and the number of constraints, respectively. Problem parameters are also defined in this part, which are described by the comment lines following each parameter.

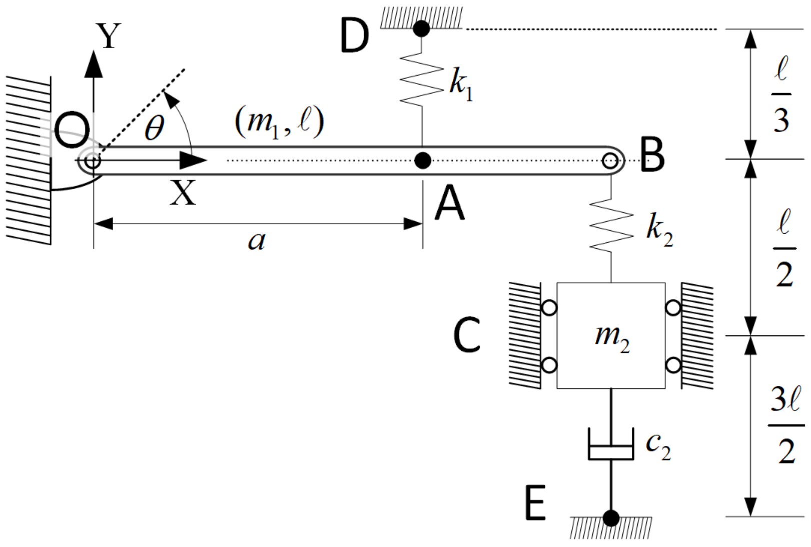


Figure A.1: Copy machine—schematic.

Listing A.1: DAETS code for the Copy Machine mechanism—preamble.

```cpp
#include <fstream>
#include <math.h>
#include <string>
#include "DAEsolver.h"
#include "ke2d.h"
#include "lagrangesetup.h"
using namespace daets;
using namespace std;
int num_q = 3, // number of generalized coordinates
    num_c = 1; // number of constraints
// Problem parameters
double g = 9.81, // gravity acceleration
    m_rod = 1,   // rod OB mass
    m_box = 1,   // box C mass
    L = 1,       // rod OB length
    w = 0.02,    // rod OB width
    h = 0.02,    // rod OB height
    a = 0.7 * L, // distance from A to O
    k1 = 100,    // AD spring constant
    k2 = 100,    // BC spring constant
    c2 = 50;     // CE damper constant
```

Listing A.2 contains the template function `CopyMachine` that encodes the DAE system of the copy machine mechanism through defining the various quantities from Equation (1.2.1). The classes used in this function are

- `BAD<T>`, a class from the FADBAD++ automatic differentiation package [5],
- `vecBAD<T>`, a `vector` whose members are of template type `BAD<T>`,
- `vec2<T>`, a `vector` containing two components of template type `T`, used to define the position/velocity of a PV as a pair of Cartesian components (x, y). `vec2<BAD<T>>` represents moving PVs in the WF, while `vec2<double>` has constant components and is used to define (a) PVs that are fixed in space, or (b) the coordinates of moving PVs in a local frame.

The PVs defined in lines 8 and 11 correspond to those shown in Figure A.1. The variable `btheta` is the angle between `OB` and the global x-axis. This angle is computed by integrating Equation (5.3.1), which is supplied to DAETS as an extra ODE via `f[4]` at line 32 (see Section 3.2).

Variable `I` holds the rod's moment of inertia about the center of mass, and `centroid` is the position of the center of mass in the rod's local frame, introduced in Section 5.3.2. The rotation matrix of the rod from (2.3.1) is calculated in line 21. The kinetic (Equation (5.3.2)) and potential (Equation (5.3.3)) energies of the mechanism are accumulated in `KE` and `PE`, respectively. Lines 22 through 28 calculate the energy terms for different elements according to the aforementioned equations. Finally, the Lagrangian is computed and stored in `Lag`.

`Rayleigh` holds the dissipation function (5.3.4) pertaining to the damper `CE`. The vector `constraints` contains the single constraint equation of the system given by Equation (5.3.5). In the end, the Lagrangian, Rayleigh's function, the state variables and generalized coordinates, as well as the constraints are passed to the function `setupEquations` which starts the Lagrangian facility and takes care of applying time and partial derivatives to generate a DAE system solvable by DAETS.

Listing A.2: DAETS code for the Copy Machine mechanism—DAE system encoding.

```
template <typename T>
void CopyMachine(T t, const T *x, T *f, void *param)
{
  vecBAD<T> q(num_q),      // generalized coordinates vector q
      qdot(num_q),         // generalized velocities vector q̇
      constraints(num_c); // constraint equations
  init_q_qp(x, q, qdot);  // initializes the the vectors defined above
  vec2<double> O(0., 0.), D(a, L / 3), E(L, -2 * L); // fixed points
  BAD<T> Lagrangian, Rayleigh; // Lagrangian and Rayleigh's dissipation function
  {
    vec2<BAD<T>> B(q[0], q[1]), C(L, q[2]), A(a / L * B), // moving PVs positions
        Bdot(qdot[0], qdot[1]), Cdot(0., qdot[2]);        // moving PV velocities
    mx2ad<T> Rot, // rod rotation matrix Q defined in Eq. (2.3.1))
        Rotp;     // rod rotation matrix time derivative Q̇
    BAD<T> btheta(x[4]);                       // angle θ between OB and global x-axis (obtained
    by integrating Eq. 5.3.1))
    BAD<T> thetadot = thetadotfcn(B, Bdot, L); // θ̇, computed in Eq. (5.3.1)
    BAD<T> KE = 0, PE = 0;                     // declare the kinetic and potential energy terms
    // Rod OB
    double I = (1. / 12) * m_rod * (SQR(L) + SQR(w));    // rod moment of inertia about its
    center of mass
    vec2<double> centroid(L / 2, 0.);                    // position of rod's center of mass in
    its local frame
    compQ_Qp(L, O, B, O, Bdot, Rot, Rotp);               // computes Q and Q̇ for rod OB
    KE += rigidBodyKE(centroid, m_rod, I, O, Rot, Rotp); // computes the kinetic energy of rod
    OB
    PE += RigidPE(m_rod, g, B.y / 2.);                   // computes the potential energy of rod
    OB
    // Box
    KE += PointMassKE(m_box, Cdot); // computes the kinetic energy of the box
    PE += RigidPE(m_box, g, C.y);   // computes the potential energy of the box
    // Springs
    PE += SpringPE(k1, L / 3., A, D) + SpringPE(k2, L / 2., B, C); // computes the potential
    energy of the springs
    Lagrangian = KE - PE; // Lagrangian
    Rayleigh = 0.5 * (c2 * SQR(Cdot)); // Rayleigh's dissipation function
    constraints[0] = LengthConstraint(B - O, L);
    f[4] = Diff(x[4], 1) - thetadot.x(); // extra ODE for computing θ̇
  }
  setupEquations(Lagrangian, Rayleigh, x, q, qdot, constraints, f); // invokes the Lagrangian
  facility to generate the DAE system (3.2.1)
}
```

The function `main` is the entry point for DAETS's execution process. As shown in Listing A.3, here we modify the simulation and output parameters and then integrate the model up to `tend`, which is the total simulation time received as a command line argument. `Solver` holds the DAE system previously built in the function `CopyMachine`. The object `opts` contains the output specifications, such as the number of output digits (via `opts.prec`), and column header names in the output file containing the numerical results (via the integer to string map `opts.varname`). the indices in line correspond to variable indices in the state vector. For example, B_x is the first member of the **q** vector, so it has index 0 as C++ has a zero-based indexing system. We also instruct DAETS to produce results for output variables only up to their first derivative in lines 13 and 14. Lastly, the integration process is carried out for four different tolerance values, given as exponents of 10 via the integer vector `pws`.

A DAETS solution object `x` holds the integration output. We set the initial values in lines 20 through 25. Finally, integration commences with a call to the `integrate` function in line 33, which writes the numerical results to the output file `outfile`.

Listing A.3: DAETS code for the Copy Machine mechanism—simulation and output setup.

```
int main(int argc, char **argv)
{
  const int n = num_q + num_c + 1; // DAE size = # gen. coords. + # constraints + 1 (for $\theta$)
  double t0 = 0.0,                 // simulation start time
      tend = 10.0;                 // simulation end time
  DAEsolver Solver(n, DAE_FCN(CopyMachine), params); // DAETS solver object
  Solver.printDAEtableau();                          // display info from structural analysis (see Section 3.3)
  OutputOptions opts;                                             // object that holds the output customizations (see Section 3.4)
  opts.prec = 16;                                                 // number of solution digits to print
  opts.heading = true;                                            // print output variable names at the top of solution file?
  opts.varname = {{0, "Bx"}, {1, "By"}, {2, "Cy"}, {4, "Theta"}}; // output variable names
  for (int i = 0; i < num_q; i++)
    opts.var_derivs.push_back({i, 1});       // only print position and velocity solutions for $\mathbf{q}$ components
  opts.var_derivs.push_back({num_q + 1, 1}); // only print position and velocity solutions for $\theta$
  // Run the model for mutiple error tolerances
  vector<int> pws{-6, -8, -10, -12}; // error tolerance exponents
  for (int i = 0; i < 1; i++)
  {
    DAEsolution x(Solver); // DAETS solution object
    // Initial conditions
    x.setT(t0)
        .setX(0, 0, L)        .setX(0, 1, 0.) // $B_x$, $\dot{B}_x$
        .setX(1, 0, 0.)       .setX(1, 1, 0.) // $B_y$, $\dot{B}_y$
        .setX(2, 0, -L / 2.)  .setX(2, 1, 0.) // $C_y$, $\dot{C}_y$
        .setX(4, 0, 0.);                      // $\theta$
    double tol = pow(10, double(pws[i])); // tol $= 10^{(\text{pws[i]})}$
    Solver.setTol(tol);                   // set absolute and relative error tolerance to tol
    std::string probName(argv[0]);                           // set probName to "copyMachine"
    std::string outName = makeOutFileName(probName, pws[i]); // set the solution file name to "copyMachine_1e<pws[i]>.out"
    std::ofstream outfile(outName + ".out");                 // set the solution statistics file name to "copyMachine_1e<pws[i]>.stat"
    x.setOutput(opts, outfile);
    SolverExitFlag flag;            // a flag indicating whether or not integration was successful
    Solver.integrate(x, tend, flag); // integrate the DAE system up to tend and store the solution in x
    if (flag != success)
      printSolverExitFlag(flag);
    x.printSolution(); // print the solution at tend to console
    outfile.close();
  }
```

Bibliography

[1] E. R. Andersen, C. Sandu, and S. Southward, *Multibody dynamics modeling and system identification of a quarter-car test rig with McPherson strut suspension*, PhD thesis, Virginia Tech., 2007. viii, 54, 55, 57, 59

[2] J. Angeles, *Rational kinematics*, Springer-Verlag, 1988. 64, 67

[3] J. Baumgarte, *Stabilization of constraints and integrals of motion in dynamical systems*, Computer Methods in Applied Mechanics and Engineering, 1 (1972), pp. 1–16. 13

[4] E. Bayo, J. G. de Jalón, A. Avello, and J. Cuadrado, *An efficient computational method for real time multibody dynamic simulation in fully cartesian coordinates*, Computer Methods in Applied Mechanics and Engineering, 92 (1991), pp. 377–395. 10

[5] C. Bendtsen and O. Stauning, *FADBAD, a flexible C++ package for automatic differentiation*, (1996). 28, 95

[6] K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, 1995. 11, 29

[7] J. Cardenal, J. Cuadrado, P. Morer, and E. Bayo, *A multi-index variable time step method for the dynamic simulation of multibody systems*, International Journal for Numerical Methods in Engineering, 44 (1999), pp. 1579–1598. 12

[8] V. Cossalter and R. Lot, *A motorcycle multi-body model for real time simulations based on the natural coordinates approach*, Vehicle System Dynamics, 37 (2002), pp. 423–447. 13

[9] J. Cuadrado, J. Cardenal, P. Morer, and E. Bayo, *Intelligent Simulation of Multibody Dynamics: Space-State and Descriptor Methods in Sequential and Parallel Computing Environments*, Multibody System Dynamics 2000 4:1, 4 (2000), pp. 55–73. 12

[10] J. G. De Jalón, *Twenty-five years of natural coordinates*, Multibody System Dynamics 2007 18:1, 18 (2007), pp. 15–33. viii, 8, 9, 10

[11] J. G. de Jalón, M. A. Serna, and R. Avilés, *Computer method for kinematic analysis of lower-pair mechanisms—I velocities and accelerations*, Mechanism and Machine Theory, 16 (1981), pp. 543–556. 10

[12] J. G. de Jalon, M. A. Serna, F. Viadero, and J. Flaquer, *A Simple Numerical Method for the Kinematic Analysis of Spatial Mechanisms*, Journal of Mechanical Design, 104 (1982), pp. 78–82. 10

[13] J. G. De Jalón, J. Unda, and A. Avello, *Natural coordinates for the computer analysis of multibody systems*, Computer Methods in Applied Mechanics and Engineering, 56 (1986), pp. 309–327. 10

[14] J. Diebel, *Representing attitude: Euler angles, unit quaternions, and rotation vectors*, Matrix, 58 (2006), pp. 1—-35. 18

[15] J. García de Jalón and E. Bayo, *Kinematic and Dynamic Simulation of Multibody Systems*, Mechanical Engineering Series, Springer New York, New York, NY, 1994. 2, 3, 4, 6, 7, 8, 24, 25

[16] H. Goldstein, C. Poole, and J. Safko, *Classical Mechanics [Textbook]*, Book, (2007), pp. 1–646. 4, 5, 6, 84

[17] M. González, D. Dopico, U. Lugrís, and J. Cuadrado, *A benchmarking system for MBS simulation software: Problem standardization and performance measurement*, Multibody System Dynamics 2006 16:2, 16 (2006), pp. 179–190. viii, 38, 43, 64, 65, 71

[18] M. González, F. González, A. Luaces, and J. Cuadrado, *A collaborative benchmarking framework for multibody system dynamics*, Engineering with Computers 2009 26:1, 26 (2009), pp. 1–9. viii, 38, 43, 64, 65, 71

[19] M. R. Hestenes, *Multiplier and gradient methods*, Journal of Optimization Theory and Applications 1969 4:5, 4 (1969), pp. 303–320. 12

[20] J. Kalirai, *Scientific discovery with the James Webb Space Telescope*, https://doi.org/10.1080/00107514.2018.1467648, 59 (2018), pp. 251–290. 1

[21] C. Kraus, *Efficient Object-Oriented Modelling, Simulation and Parameter Estimation for Biomechanical Problems*, (2006). 12

[22] C. Kraus, H. G. Bock, and H. Mutschler, *Parameter Estimation for Biomechanical Models Based on a Special Form of Natural Coordinates*, Multibody System Dynamics 2005 13:1, 13 (2005), pp. 101–111. 12

[23] C. Kraus, M. Winckler, and H. G. Bock, *Modeling Mechanical DAE Using Natural Coordinates*, Mathematical and Computer Modelling of Dynamical Systems, 7 (2001), pp. 145–158. 12

[24] A. Lago, D. Trabucco, and A. Wood, *Case studies of tall buildings with dynamic modification devices*, Damping Technologies for Tall Buildings, (2019), pp. 533–919. 25

[25] W. M. Lioen and J. J. de Swart, *Test Set for Initial Value Problem Solvers*, Centrum voor Wiskunde en Informatica, (1998). 36

[26] Mathworks, *Simulation and Model-Based Design*, 2020. 34

[27] ——, *Simscape Multibody*, 2021. 34

[28] MATLAB (The Mathworks Inc.), *MATLAB*, 2020. 34

[29] N. M. U. Nedialkov and J. Pryce, *Multibody Dynamics in Natural Coordinates through Automatic Differentiation and High-Index DAE Solving*, tech. rep., McMaster Univeristy. 1, 4, 34, 35, 49

[30] N. S. Nedialkov and J. Pryce, *DAETS user guide.*, tech. rep., McMaster University, Hamilton, ON, 2013. viii, 1, 13, 29, 30, 31

[31] N. S. Nedialkov and J. D. Pryce, *Solving Differential-Algebraic Equations by Taylor Series (I): Computing Taylor Coefficients*, BIT Numerical Mathematics 2005 45:3, 45 (2005), pp. 561–591. 13, 30

[32] ——, *Solving differential-algebraic equations by Taylor series (II): Computing the System Jacobian*, BIT Numerical Mathematics, 47 (2007), pp. 121–135. 30

[33] ——, *Solving differential algebraic equations by taylor series (III): The DAETS code*, Journal of Numerical Analysis, Industrial and Applied Mathematics, 3 (2008). 30

[34] P. E. Nikravesh, *An Overview of Several Formulations for Multibody Dynamics*, Product Engineering: Eco-Design, Technologies and Green Energy, (2004), pp. 189–226. 6, 7

[35] B. Paul and D. Krajcinovic, *Computer Analysis of Machines With Planar Motion: Part 1—Kinematics*, Journal of Applied Mechanics, 37 (1970), pp. 697–702. 7

[36] R. L. Pio, *Euler Angle Transformations*, IEEE Transactions on Automatic Control, 11 (1966), pp. 707–715. 8

[37] J. D. Pryce, *A simple structural analysis method for DAES*, BIT Numerical Mathematics, 41 (2001), pp. 364–394. 29

[38] J. D. Pryce and N. S. Nedialkov, *Multibody dynamics in natural coordinates through automatic differentiation and high-index DAE solving*, Acta Cybernetica, 24 (2020), pp. 315–341. 14, 17

[39] J. D. Pryce, N. S. Nedialkov, G. Tan, and X. Li, *How AD can help solve differential-algebraic equations*, https://doi.org/10.1080/10556788.2018.1428605, 33 (2018), pp. 729–749. 13, 39

[40] R. R. Ryan, *ADAMS — Multibody System Analysis Software*, Multibody Systems Handbook, (1990), pp. 361–402. 11

[41] M. A. Serna, R. Avilés, and J. García de Jalón, *Dynamic analysis of plane mechanisms with lower pairs in basic coordinates*, Mechanism and Machine Theory, 17 (1982), pp. 397–403. 10

[42] L. F. Shampine, *Tolerance proportionality in ODE codes*, (1989), pp. 118–136. 30, 36

[43] P. N. Sheth and J. J. Uigker, *IMP (Integrated Mechanisms Program), A Computer-Aided Design Analysis System for Mechanisms and Linkage*, Journal of Engineering for Industry, 94 (1972), pp. 454–464. 7

[44] R. C. Smith and E. J. Haug, *DADS — Dynamic Analysis and Design System*, Multibody Systems Handbook, (1990), pp. 161–179. 13

[45] VILALLONGA and G., *Numerical Kinematic Analysis of Three-dimensional Mechanism Using a "Natulral" Systme of Lagrange Coordinates*, ASME paper, 2 (1985). 10

[46] R. von Schwerin, *MultiBody System SIMulation: Numerical Methods, Algorithms, and Software - Reinhold von Schwerin - Google Books*, 1999. 4, 7, 8, 10, 11, 80, 84

[47] A. Wächter and L. T. Biegler, *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*, Mathematical Programming 2005 106:1, 106 (2005), pp. 25–57. 30

[48] U. Wiki, *Windows Subsystem for Linux*, 2022. 36