WARSAW UNIVERSITY OF TECHNOLOGY

MECHANICAL ENGINEERING
ENGINEERING AND TECHNOLOGY

Ph. D. Thesis

Paweł Wojciech Maciąg, M. Sc.

Optimal Control of Multibody Systems Using an Efficient, Parallelizable Adjoint Method

Supervisor

Paweł Malczyk, Ph. D., D. Sc.

Additional Supervisor

Łukasz Woliński, Ph. D.

WARSAW 2023

Drogiej Marysi,
moim Rodzicom

ACKNOWLEDGEMENTS

First, I would like to express my sincerest gratitude to D.Sc., Ph.D. Paweł Malczyk for his countless advice, genuine support, and all the effort put into our collaborative work during the past several years. I am also thankful to everyone collaborating with me in the Faculty of Power and Aeronautical Engineering. Special thanks to professor Marek Wojtyra, head of the Division of Theory of Machines and Robots, professor Janusz Frączek, dean of the Faculty of Power and Aeronautical Engineering, and Ph.D. Łukasz Woliński for reviewing the thesis. Thanks should also go to M.Sc. Maciej Pikuliński for helping me implement a substantial part of the benchmark program in chapter 6. Finally, I am deeply indebted to the National Science Center [1] as well as to the Warsaw University of Technology [2] for generous financial remuneration.

Last but not least, this endeavor would not have been possible without the support of Maria – my wonderful fiancé – as well as my parents Grażyna and Andrzej. Thank you for your relentless support and profound belief in my work.

Warsaw, May 2023

Paweł Wojciech Maciąg

[1] OPUS 15 under grant no. 2018/29/B/ST8/00374

[2] Excellence Initiative, Research University in the field of Artificial Intelligence and Robotics under grant no. 1820/25/Z01/POB2/2021

ABSTRACT

The importance of efficient computational methods for computer-aided design and control of robotic systems has significantly increased over the last decade. Optimal control (OC) is a well-established field of knowledge and a universal tool for modern non-linear problems drawing significantly from optimization methods. Concurrently, the optimization-based approach relies on algorithms for accurate, systematic, and efficient calculation of derivatives with respect to the design variables.

The adjoint method – derived from the optimal control theory – uses a multi-body system (MBS) dynamics model to compute the performance measure's gradient efficiently and is a natural approach in the presence of a large number of design variables. The mathematical model of the system constitutes the equations of motion, which are herein expressed via Hamiltonian formalism. *The first contribution of this dissertation is the unification of the adjoint method with constrained Hamilton's equations in mixed absolute and joint coordinates.*

Although methods for solving optimal control problems became relatively mature, there is a need to develop faster and more efficient algorithms that can be applied in practice. An important issue is their scalability against the problem size, e.g., the number of bodies in the kinematic chain. *Another contribution of this work lies in improving the adjoint method by developing a parallel algorithm for solving the forward and adjoint problems based on the divide-and-conquer (DCA) scheme.* The core of the proposed method lies in a recursive computation of the adjoint system's coefficients, based on a binary tree related with the topology of the MBS. Since computational procedures associated with each of the graph nodes can be executed in parallel, the DCA approach is highly scalable when working on a sufficiently large number of computing nodes.

The divide-and-conquer algorithm has already been used to solve the equations of motion (in both Lagrangian and Hamiltonian formulations) and to compute the gradient of the performance measure via the direct differentiation method. *This dissertation extends*

the current state-of-the-art by proposing a novel Hamiltonian Adjoint-based Divide-and-Conquer-Algorithm (HADCA), which enables solving the adjoint problem by performing parallel computations.

The developed methods were tested on various numerical examples in a Matlab environment to present their application possibilities. Several optimal design and control problems for planar and spatial mechanisms with different topologies of the kinematic chain (open and closed) have been solved. Obtained results illustrate the correctness of the proposed methods. Finally, the results were discussed in detail regarding the solutions' accuracy and efficiency.

The scalability of HADCA was tested by solving the benchmark optimal control problem with a MBS characterized by a variable (adjustable) number of degrees of freedom in the kinematic chain. To this end, the developed method was implemented in C++, and the computations were parallelized with OpenMP directives. The benchmark model was employed to check the developed formulation's correctness and implementation. Execution times were also measured for various conditions, where the varied parameters were the number of bodies (i.e., problem size) and the number of available threads.

The developed adjoint method was also employed in a practical control system of a parallel, planar, two-degree-of-freedom robot, for which a mathematical model has been defined. Computed input signals were fed to motors as a feedforward compensation that acted in conjunction with the feedback loop. Control performance and error magnitudes were tested during the execution on the hardware when the robot was tasked to perform complex trajectories with possible constraints imposed on the control signal.

Keywords: adjoint method, multibody dynamics, sensitivity analysis, optimal control, Hamiltonian dynamics, optimization methods

STRESZCZENIE

W ciągu ostatnich dekad znaczenie efektywnych obliczeniowo metod w komputerowo wspomaganym projektowaniu oraz sterowaniu układami robotycznymi wyraźnie wzrosło. Sterowanie optymalne jest dobrze ugruntowaną dziedziną wiedzy dającą narzędzia do rozwiązywania współczesnych nieliniowych problemów dynamiki i czerpiącą znacząco z metod optymalizacji. Jednocześnie wiele spośród nich wymaga zastosowania efektywnych algorytmów do systematycznego i dokładnego wyznaczania pochodnych względem zestawu zmiennych decyzyjnych.

Metoda adjoint – wywodząca się z teorii sterowania optymalnego – wykorzystuje model dynamiki układu wieloczłonowego (UW) do efektywnego obliczenia gradientu wskaźnika jakości i jest naturalnym podejściem w przypadku obecności dużej liczby zmiennych decyzyjnych. Model matematyczny układu ma wtedy formę równań ruchu, które w niniejszej rozprawie są wyrażone za pomocą formalizmu Hamiltona. *Pierwszym osiągnięciem rozprawy jest połączenie metody adjoint z hamiltonowskimi modelami UW sformułowanymi w mieszanych współrzędnych złączowych i absolutnych.*

Choć metody służące do rozwiązywania zagadnień sterowania optymalnego charakteryzują się dużym stopniem dojrzałości, wciąż istnieje potrzeba rozwoju coraz efektywniejszych algorytmów możliwych do zastosowania w praktyce. Kluczową kwestią jest również ich skalowalność ze względu na rozmiar problemu, np. liczbę członów w łańcuchu kinematycznym. *Istotnym celem niniejszej rozprawy jest usprawnienie metody adjoint przez opracowanie algorytmu równoległego rozwiązującego zadanie proste dynamiki i problem sprzężony (adjoint) w oparciu o schemat Dziel i Zdobywaj (DiZ).* Rdzeniem proponowanej metody jest możliwość rekurencyjnego wyznaczania współczynników dla problemu sprzężonego, która bazuje na drzewie binarnym stowarzyszonym z topologią UW. Poszczególne procedury obliczeniowe związane z węzłami grafu można wykonywać w sposób równoległy, dzięki czemu podejście DiZ odznacza się dobrą skalowalnością obliczeń na wielu procesorach

Algorytm DiZ stosowano już do rozwiązywania równań ruchu (zarówno w formie lagranżowskiej, jak i hamiltonowskiej) oraz do obliczania gradientu wskaźnika jakości metodą różniczkowania bezpośredniego (ang. direct differentiation method). *Niniejsza rozprawa poszerza obecny stan wiedzy o propozycję metody adjoint zaimplementowanej w schemacie DiZ i umożliwiającej wykorzystanie obliczeń równoległych.*

Opracowane metody poddano zróżnicowanym testom numerycznym w środowisku Matlab i pokazano ich możliwości aplikacyjne. Rozwiązano wybrane problemy projektowania oraz sterowania optymalnego dla mechanizmów płaskich i przestrzennych charakteryzujących się różnymi topologiami łańcucha kinematycznego. Otrzymane wyniki potwierdzają poprawność opracowanych metod. Ostatecznie w rozprawie szczegółowo omówiono otrzymane rezultaty pod kątem jakości rozwiązań oraz efektywności obliczeń.

Skalowalność metody adjoint i algorytmu DiZ zbadano poprzez rozwiązanie problemu sterowania optymalnego dla UW charakteryzującego się możliwością relatywnie łatwej zmiany liczby stopni swobody w łańcuchu kinematycznym. W tym celu opracowane metody zaimplementowano w C++, a do zrównoleglenia obliczeń zastosowano dyrektywy kompilatora OpenMP. W oparciu o testowy model UW sprawdzono poprawność sformułowania oraz samej implementacji. Zmierzono również czasy wykonania programu dla różnych warunków zadania, w których zmianom podlegała liczba członów oraz liczba dostępnych wątków.

Opracowana metoda adjoint została zastosowana w układzie sterowania rzeczywistym robotem równoległym (planarny robot o dwóch stopniach swobody), dla którego zbudowano i zidentyfikowano model matematyczny. Wyznaczone sygnały sterujące podano jako sprzężenia kompensujące w przód typu feedforward, które działały w koniunkcji z sygnałami z pętli sprzężenia zwrotnego. Badaniom podlegała jakość sterowania obiektem oraz wartości uchybów regulacji w przypadku skomplikowanych trajektorii zadaniowych robota oraz potencjalnych ograniczeń na amplitudy sygnału sterującego.

Słowa kluczowe: metoda adjoint, dynamika układów wieloczłonowych, analiza wrażliwości funkcjonału, sterowanie optymalne, metody optymalizacji

CONTENTS

LIST OF SYMBOLS

The list describes several symbols that have been used within this thesis. In this work, the "is in" sign $\in$ is used to specify the size of a matrix, whereas vectors are treated as single-column matrices, and the symbol $\mathcal{R}$ denotes the space of real numbers. The *global formulation* section involves only quantities tied to a formulation based on the angular velocity.

Dynamic Equations: global formulation

$\mathbf{q} \in \mathcal{R}^{n_q}$	Absolute coordinates vector
$\boldsymbol{\alpha} \in \mathcal{R}^{n_\alpha}$	Joint coordinates vector
$\mathbf{v} \in \mathcal{R}^{n_v}$	Absolute velocity vector
$\boldsymbol{\beta} \in \mathcal{R}^{n_\beta}$	Joint velocity vector
$\mathbf{p} \in \mathcal{R}^{n_v}$	Physical momenta vector
$\mathbf{p}^* \in \mathcal{R}^{n_v}$	Augmented momenta vector
$\widehat{\mathbf{p}} \in \mathcal{R}^{n_\beta}$	Joint momenta vector
$\mathbf{f} \in \mathcal{R}^{n_v}$	External force vector
$\boldsymbol{\sigma} \in \mathcal{R}^{m}$	Impulses of contraint reactions
$\boldsymbol{\lambda} \in \mathcal{R}^{m}$	Constraint reaction forces
$\boldsymbol{\Phi} \in \mathcal{R}^{m}$	Constraints vector
$\boldsymbol{\Phi}_{\mathbf{q}} \in \mathcal{R}^{m \times n_q}$	Constraints Jacobian matrix
$\mathbf{C} \in \mathcal{R}^{m \times n_v}$	Constraints Jacobian matrix

$\mathbf{H} \in \mathcal{R}^{n_v \times n_\beta}$ Global motion subspace matrix

$\mathbf{M} \in \mathcal{R}^{n_v \times n_v}$ Mass matrix of the multibody system

Dynamic Equations: local formulation

$\mathbf{H}_i \in \mathcal{R}^{6 \times N^{\text{dof}_i}}$ Motion subspace of joint i

$\mathbf{D}_i \in \mathcal{R}^{6 \times (6 - N_i^{\text{dof}})}$ Constrained velocity subspace of joint i

$\mathbf{M}_1^A \in \mathcal{R}^{6 \times 6}$ Mass matrix computed at handle 1 of body A

$\mathbf{V}_1^A \in \mathcal{R}^6$ Spatial velocity vector at handle 1 of body A

$\mathbf{P}_2^B \in \mathcal{R}^6$ Spatial physical momenta vector at handle 2 of body B

$\overline{\mathbf{P}}_2^A \in \mathcal{R}^6$ Spatial articulated momenta vector at handle 2 of body A

$\mathbf{F}_1^B \in \mathcal{R}^6$ Spatial active force vector at handle 1 of body B

$\mathbf{T}_1^A \in \mathcal{R}^6$ Impulses of constraint loads at handle 1 of body A

$\mathbf{Q}_1^A \in \mathcal{R}^6$ Spatial accumulated force vector at handle 1 of body A

$\overline{\mathbf{Q}}_1^A \in \mathcal{R}^6$ Spatial articulated force vector at handle 1 of body A

$\boldsymbol{\zeta}_{ij}^A \in \mathcal{R}^{6 \times 6}$ Propagation coefficient matrix of the EOM for body A

$\boldsymbol{\zeta}_{i0}^A \in \mathcal{R}^6$ Propagation coefficient vector of the EOM for body A

Adjoint equations

$\boldsymbol{\eta}, \boldsymbol{\xi} \in \mathcal{R}^{n_v}$ Absolute adjoint variables

$\boldsymbol{\mu} \in \mathcal{R}^m$ Adjoint "reactions" Lagrange multipliers

$\widehat{\boldsymbol{\eta}}, \widehat{\boldsymbol{\xi}} \in \mathcal{R}^{n_\beta}$ Joint-space adjoint variables

$\boldsymbol{\chi}_{i0}^A \in \mathcal{R}^6$ Propagation coefficient vector of the adjoint system for body A

Other symbols

$\mathbf{u}(t) \in \mathcal{R}^{n_u}$ Vector of input functions to the multibody system

$\mathbf{b} \in \mathcal{R}^k$ Vector of design variables or discretized inputs signals

N_b, `Nbodies` Number of bodies in the multibody system

Δt Time step or discretization step

LIST OF ACRONYMS

COM Center of Mass

DAE Differential Algebraic Equation

DAQ Data Acquisition

DCA Divide-and-Conquer Algorithm

DDM Direct Differentiation Method

HADCA Hamiltonian Adjoint-based Divide-and-Conquer Algorithm

HDCA Hamiltonian Divide-and-Conquer Algorithm

EOM Equations of Motion

MBS Multibody System

MPC Model Predictive Control

NLP Non-Linear Programming

OC Optimal Control

OD Optimal Design

ODE Ordinary Differential Equation

RHS Right-hand Side

CHAPTER 1

INTRODUCTION

1.1 Motivation

Since the famous brachistochrone problem had been formulated and solved in the 17$^{\text{th}}$ century [48], the optimal control (OC) theory has been continuously developed, utilized in new engineering fields, and applied to more complex or practical problems. The main objective of OC is to determine the input control signals that will cause a process to satisfy given constraints while minimizing specific performance criteria.

Reliable and efficient optimal control methods are of paramount importance for dynamical systems of tomorrow including aerospace (e.g. aerial motion planning [26]), automotive (e.g. self-driving cars [47]), biomechanical [117] or robotic (e.g. manipulator or nonholonomic motion planning [39, 119, 126]) applications that more and more often work autonomously in a dynamic environment. Although the classical and optimal control algorithms have been developed and well understood for over half of a century [51], the need for their rapid execution is ever-growing. Some practical applications of optimal control theory include the design of robots [33] and linkages [82], balancing of mechanisms [118], control of compliant [7] or grasping mechanisms [102], underactuated systems' control [5], and mission planning algorithms [49, 105].

Consequently, the multibody dynamics community established itself in the second half of the 20$^{\text{th}}$ century [99, 112] with the rise of modern computers and the contemporary approach to modeling and solving complicated differential-algebraic equations (DAEs). Once it was possible to simulate the dynamic response of a multibody system, the need to predict how a specific parameter could affect its performance immediately followed, giving rise to the problem of *design sensitivity analysis*.

The goal of sensitivity analysis is to evaluate derivatives of one or more expressions with respect to one or several independent quantities, known as design variables. Although there are various uses for sensitivity information, exploiting it in gradient-based optimization is the primary motivation. The computation of gradients is often the most costly step in the optimization process; hence, efficient methods that accurately calculate sensitivities are essential. Multiple approaches to sensitivity analysis have been proposed that can be ordered into different families, such as:

- black-box methods (complex-step, finite differences) [84] ,
- automatic differentiation [68],
- model-based approaches: direct differentiation and adjoint methods [30, 90],
- symbolic differentiation [40].

Each approach has its own set of benefits and drawbacks. For example, black-box methods are easy to implement; however, their performance scales badly with the number of state or design variables. On the other hand, model-based approaches can be the most efficient at the expense of embedding the underlying dynamics into the sensitivity analysis problem. In the field of multibody system (MBS) dynamics, computing the gradient of the performance measure is a bottleneck of the underlying optimal control or design problems, specifically in the presence of high-degree-of-freedom systems.

This work particularly pertains to the adjoint approach since its efficiency is not affected by the high number of design variables. The adjoint method originates from optimal control theory, where it is associated with the indirect methods of open-loop control [8]. Initially, this methodology has gained attention in the fluid dynamics community [46], whereas its first application to multibody dynamics was introduced by Edward J. Haug [57]. Recent years have delivered a surge of interest and new publications around this subject [70] which potentially can be attributed to the pursuit of more efficient and high-fidelity algorithms.

The adjoint method is based on a mathematical model of the given phenomenon, e.g., the underlying dynamics expressed by the equations of motion (EOM). Herein, the dynamic equations will be described by means of the Hamiltonian formulation with modified Lagrangian. Compared with other formulations, the chosen description is characterized by desirable numerical properties, such as higher stability of the numerical solution [29, 69, 16]. Various Hamiltonian-based formulations have been studied in the literature for the purpose of efficient dynamic analysis [9, 91]; however, few publications can be found regarding the adjoint sensitivity analysis in the specified framework [79]. This work develops such an approach and exploits the outcome in number of numerical examples and in a practical hardware implementation.

The initial redundant formulations of EOM have been characterized by $O(N_b^3)$ time complexity [6, 55], where N_b stands for the number of bodies. Subsequently, based on the recursive propagation of dynamic equations from the body to body, newer algorithms significantly improved to $O(N_b)$ (i.e., linear complexity) [43, 64, 92]. Although these solutions are CPU performant, the algorithms are sequential and most efficient for serial topologies (i.e., open-loop kinematic chains). Another family of methods has been introduced to address this issue: algorithms characterized by $O(\log(N_b))$ complexity [13, 44].

A similar pursuit after more scalable algorithms has also been visible in the field of sensitivity analysis [114]. For example, the direct differentiation method has been successfully implemented as an $O(N_b)$ [2] algorithm as well as in the form of $O(\log(N_b))$ [87] procedure. The latter case involves a paradigm known as divide-and-conquer, which is based on a recursive traverse along a binary tree and later will be explained in greater detail. On the other hand, the number of publications about efficient and parallelizable adjoint sensitivity analysis implementations is not abundant [65]. This thesis aims to expand the current state-of-the-art in this niche by proposing a parallelizable algorithm for the adjoint method based on the divide-and-conquer paradigm. The proposed approach will be derived in detail and tested against multiple examples in the simulation and real environment.

1.2 Thesis structure

This thesis is organized as follows. **Chapter 2** presents the scope of this thesis and puts it in the perspective of the related work. The reader is introduced to the notation used throughout this study. The optimization problem is formulated and various methods for obtaining gradient information are depicted. Ultimately, thesis objectives and contributions are listed in the last section of the chapter.

The equations of motion for multi-rigid-body systems formulated in redundant and minimal sets of coordinates are derived in **Chapter 3**. Two complementary sub-spaces corresponding to the joint's motion and constrained directions are introduced. Finally, the Hamiltonian Divide-and-Conquer (HDCA) algorithm is established for a forward dynamics problem. Specifically, the chapter presents two detailed derivations allowing for recursive evaluation of the MBS velocities (sec. 3.4.1) and the derivative of momenta (sec. 3.4.2), respectively.

Next, in **chapter 4**, the adjoint method tailored to the constrained Hamilton's equations of motion is presented. Subsequent sections in the chapter introduce the concept of independent adjoint variables and present the system adjoint to dynamic equations

formulated as ordinary differential equations (ODEs). The relationship between DAE and ODE formulations of the equations of motion is studied, which allows for defining spatial relations between adjoint variables as well as building dependencies between absolute and joint coordinate counterparts of the adjoint variables. The outcome of these studies allows us to derive the Hamiltonian Adjoint-based Divide-and-Conquer-Algorithm (HADCA), which is presented in the final section.

Chapter 5 presents various problems solved with the aid of the proposed methods and reports on the encountered challenges. Multiple examples are investigated, for example, optimal design or optimal control of planar and spatial multibody systems. The contents vary from simple test cases to practical problems that can be implemented in the laboratory setting.

The methods developed in chapter 4 have been implemented, on a shared memory parallel computer. **Chapter 6** presents the results of code benchmarking that tests the scalability and measures the execution time for various cases of the adjoint sensitivity analysis. Furthermore, this chapter delivers detailed algorithms that put most of the relations developed earlier in a broader perspective.

Consequently, **chapter 7** pertains to performing an experiment on a five-bar MBS actuated by two DC motors. First, the optimal control problem is solved offline (based on the electromechanical model of the device) via the adjoint method, which generates the reference trajectory of the end-effector as well as the appropriate input voltages. In the following part, these results are fed to the system as a feedforward control signal, which supplies the feedback control loop. The outcomes gathered from multiple experiments comparing the efficiency and accuracy of the proposed algorithms are delivered at the end.

Ultimately, **chapter 8** presents a summary of the thesis, where each of the prescribed herein goals are summarized. **Appendices A and B** are devoted to explaining the derivations presented in chapter 4 in greater detail, whereas **appendix C** contains supplementary pseudocode that complements algorithms presented in chapter 6.

CHAPTER 2
STATE-OF-THE-ART

2.1 Introduction

The scope of this thesis interconnects the fields of multibody systems' dynamics, indirect methods of optimal control, and algorithms for parallel computations. In this literature review, we will investigate other approaches to MBS optimization and trajectory generation studied currently by the researchers. The optimal control approach is inseparably interconnected with sensitivity analysis, which is a broad field of research in itself. In the following part of this chapter, various methods of gradient calculation will be outlined. The most performant family of methods, known as model-based approaches, will be explained in greater detail. Finally, based on the literature review and state-of-the-art, the central thesis of this work will be presented, supplied by a set of intermediate goals it shall fulfill.

2.2 Literature review

Let us begin by placing the adjoint method in the general context of optimal control (see fig. 2.1). The main distinction lies in open- and closed-loop control, where we search for appropriate input control sequence in the former case and design an optimal *control policy* (based on sensory feedback) in the latter case. Furthermore, the core of the closed-loop approach involves *dynamic programming*: an iterative method for dealing with optimal control problems by solving subproblems based on the *Bellman's principle of optimality*. According to this principle, the subsolutions of an optimal solution of the general problem are themselves optimal solutions for their subproblems [66]. This observation allows for

a recursive deduction of appropriate decision from the final state backward (e.g., final time of the dynamic simulation), until the whole decision domain has been covered.

The recurrence relation based on Bellman's principle can be applied to discrete systems as well. By extending the specified principle into continuous systems, one can come up with the Hamilton-Jacobi-Bellman equation, which is a partial differential equation generalizing the optimal control methodology onto continuous dynamic systems. The dynamic programming approach is also the basis for designing a Linear Quadratic Regulator (LQR) – a local stabilizing controller that is often used as a first effective approach for a wide variety of problems [3]. This concept assumes linear dynamics and quadratic performance measure that systematically computes optimal control law for a given task. The limitation of the LQR approach reveals itself when one aims at tracking specified trajectory. The need to linearize system dynamics around whole trajectory renders this method difficult, or sometimes infeasible, to implement practically [88]

Iterative LQR (iLQR) is a technique that circumvents specified issues. This algorithm introduces a forward pass in which the nominal trajectory is updated by applying the control policy to the nonlinear dynamics [123]. Additionally, the backward pass involves the linearization of the dynamics and quadratization of the cost function around a new nominal trajectory. The procedure alternates between forward and backward passes, yielding new control law at each iteration.

A similar approach is to approximate the Bellman equation directly instead of first approximating the dynamics and the cost function. Differential dynamic programming (DDP) directly builds a quadratic approximation of the right-hand side of the Bellman equation, which may then be solved analytically [19]. While DDP yields a more accurate approximation, it requires the computation of the costly second-order derivatives of the dynamics. According to the general opinion within the robotics community, iLQR is sufficient for most applications; however, more efficient DDP implementations are under development [94].

Aside from dynamic programming, two major approaches for solving OC problems include direct and indirect methods. In the former approach, the state and control of the optimal control problem are discretized in some manner, and the problem is transcribed to a nonlinear programming problem (NLP) [11]. The NLP (i.e., nonlinear optimization problem) is then solved using well-known optimization techniques [97]. As specified earlier, indirect methods involve using the calculus of variations to determine the necessary conditions for optimality of the original optimal control problem. The indirect approach leads to a multiple-point boundary-value problem (BVP) which is usually solved via (multiple) shooting methods or collocation techniques [83].

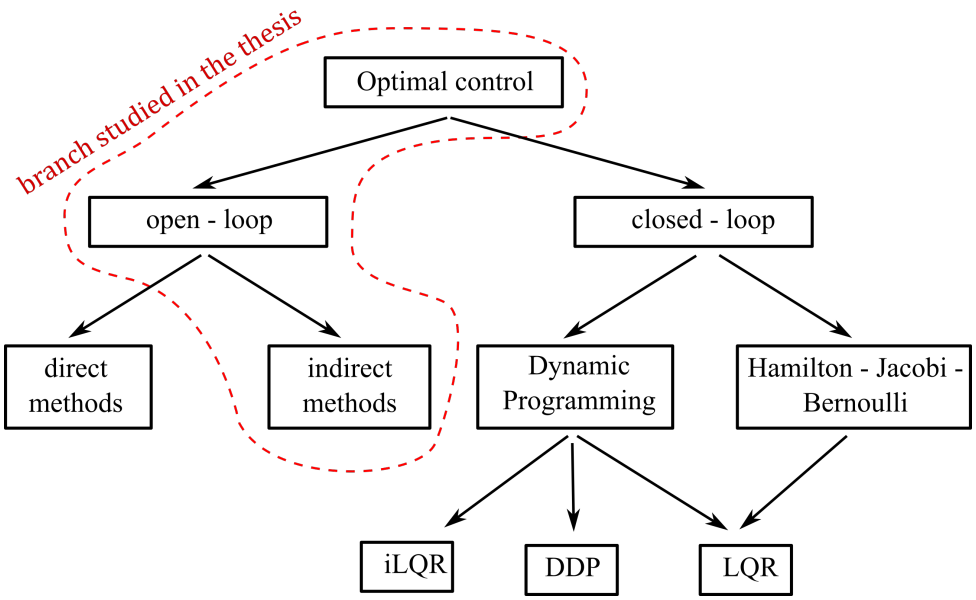


Figure 2.1: Taxonomy of different approaches for optimal control problems

For a wide range of multibody dynamics problems, it is possible to decouple the BVP into two categories: the underlying initial value problem composed of the forward dynamics problem and the adjoint problem. The former must be solved first to provide the appropriate values for the latter phase [113]. Solving the adjoint system yields quantities (adjoint variables) allowing for efficient sensitivity analysis of the performance measure. In the multibody dynamics setting, it is a common practice to formulate the EOM as a set of constrained DAEs. The resultant adjoint system has a similar structure to the organization of the equations of motion [104]. The cost of solving the adjoint system is equivalent to computational expenses spent in the forward dynamics problem, which is a favourable property of the adjoint method.

The adjoint approach represents a comprehensive computational framework, rather than a single method, where many authors develop this broad branch of research in various directions. For example, the adjoint method for solving the nonlinear optimal control problem has been applied in different engineering areas such as parameter identification [35, 67, 72], structural mechanics [58], time-optimal control problems [14, 41], and feedforward control design [76]. Furthermore, the integration routine may be combined with the discretized optimal control problem to come up with the discrete adjoint equations [71, 80]. The adjoint method has been also employed in hybrid systems involving discontinuities or switching modes [31, 127].

Software

Several efficient and versatile tools for aiding computer-based engineering have been developed so far. They can be roughly categorized as software for simulating physical environments, optimization or optimal control of MBS, and sensitivity analysis.

From a vast number of physical simulators, we can distinguish Mujoco [120] (an all-purpose physical engine recently acquired by Google Deepmind), Chrono [111], Pinocchio [25], and MBDyn [86]. The effectiveness of these programs vary with possible problem scales that can be addressed and additional capabilities, such as sensitivity analysis [25], integration with other packages, or simulation of specific phenomena (hydraulic, aerodynamic [86], complex contacts [111], etc.).

Certain simulation packages are primarily tools for solving optimal control problems. Some of these state-of-the-art optimal control software rely on direct methods of optimal control [11, 101], implicit function theorem [61], or DDP [62].

The underlying issue for most problems in the field of multibody dynamics is the efficient solution of differential-algebraic equations. To this end, a number of powerful solvers have been written [73]. Historically, we can mention such codes as DASSL, DASPK solvers, and DADS or ADAMS [99] (commercially available to this date) packages. Currently, one of the most popular general-purpose DAE solvers is *Sundials* [60]. It involves adjoint methods for carrying out a first-order sensitivity analysis with respect to model parameters or initial conditions. The software involves modern high-performance computing techniques, such as OpenMP, MPI, CUDA; however, the suite is not written directly with the multibody dynamics community in mind.

The most beneficial property of the adjoint method, which is its efficiency in problems involving a large number of design variables, renders it feasible for utilization as a practical tool for engineering problems [115]. Some solutions developed by multibody dynamic researchers with capabilities for direct and adjoint sensitivity include MBSLIM [38, 122] and Freedyn [89]. This thesis proposes a novel parallel algorithm based on the divide-and-conquer paradigm and utilizing the adjoint approach. It allows for fast gradient computation in optimal control or optimization problems. The methodology investigated in this thesis has the potential to be implemented in some of the described packages or as a tool on its own.

2.3 Optimal control problem formulation

A classical optimal control problem assumes that the sought input variables are continuous. Nevertheless, any optimal control problem may be solved via nonlinear optimization

methods, often referred to as *nonlinear programming* or simply NLP. The NLP problem can be formulated in the following way: find a set of static parameters $\mathbf{b} \in \mathcal{R}^k$ that minimize the performance measure $J(\mathbf{y}, \dot{\mathbf{y}}, \mathbf{b})$, subjected to the equality constraints $\mathbf{c}_E = \mathbf{0}, \mathbf{c}_E \in \mathcal{R}^{n_E}$.[1] Hence, a large class of multibody dynamics problems can be captured and systematically solved by defining the problem as a minimization of the given performance measure [11]:

$$J[\mathbf{y}(\cdot), \dot{\mathbf{y}}(\cdot), \mathbf{b}] = \int_0^{t_f} h(t, \mathbf{y}, \dot{\mathbf{y}}, \mathbf{b}) \, \mathrm{d}t + S(\mathbf{y}(t_f), \dot{\mathbf{y}}(t_f)), \tag{2.1}$$

subjected to the $2n + m$ equality constraints of the form:

$$\mathbf{F}(t, \mathbf{y}(t), \dot{\mathbf{y}}(t), \boldsymbol{\nu}(t), \mathbf{u}(t), \mathbf{b}) = \mathbf{0} \tag{2.2}$$

and $2n$ initial conditions:

$$\mathbf{y}(0) = \mathbf{y}_0, \qquad \dot{\mathbf{y}}(0) = \dot{\mathbf{y}}_0. \tag{2.3}$$

Symbol $\mathbf{y} \in \mathcal{R}^n$ denotes a differential state variable, whereas $\boldsymbol{\nu} \in \mathcal{R}^m$ is purely an algebraic state variable and can be associated with, e.g., reaction forces arising in the system. For brevity, equation (2.2) involves both the dynamic equations of motion and path constraints, i.e., purely algebraic equations. The integrand denoted by h represents the expression to be minimized in a fixed time horizon t_f. The term S in eq. (2.1) specifies the terminal cost of the performance measure, which is a convenient way to prescribe a particular configuration of the system at the final time.

The quantities $\mathbf{b} \in \mathcal{R}^k$ and $\mathbf{u}(t) \in \mathcal{R}^{n_u}$ denote input to the model in the form of static parameters and continuous functions, respectively. In practice, the elements of $\mathbf{u}(t)$ must be discretized to seamlessly work with NLP solvers. In the simplest case, this is realized by sampling functions $u_i(t)$, $i = \{1, \cdots, n_u\}$ with a constant time step Δt and recording these static (i.e. time-independent) values in the vector $\mathbf{b}$. Consequently, one can recreate the original signal $u_i(t)$ from the vector $\mathbf{b}$ via appropriate interpolation technique. Although combining both types of input within a single problem is possible [98], this work will assume only one type of input at a time for simplicity. Hence, from now on, by referring to variable $\mathbf{b}$, we mean either static design parameters or discretized control signals $\mathbf{u}$.

The class of problems that can be addressed by the specified formulation involves sufficiently smooth (continuous, differentiable) functions. Furthermore, the final time is fixed, which prevents us from assigning it as a design variable. The solution of problem (2.1) –

[1] A common optimal control problem involves additionally inequality constraints in form $\mathbf{c}_I \leqslant \mathbf{0}$, $\mathbf{c}_I \in \mathcal{R}^{n_I}$; however, this thesis does not assume their appearance. It is, nevertheless, possible to take this type of constraints into account by adding penalty term to the cost functional (c.f. sec. 7.6).

(2.3) is not a trivial task. Most practical problems require certain guarantees for obtaining a solution in a reasonable time [59, 83]. Therefore, a majority of approaches involves deterministic optimization techniques, which rely heavily on computing the gradient of the performance measure. Various techniques have been developed to address this issue efficiently, varying in an array of properties, such as the simplicity of implementation, accuracy, efficiency, or memory usage.

The most straightforward approach is the finite differences method, which treats the whole dynamics of a MBS as a black box and yields the sensitivities directly by perturbing the cost function about each design variable separately. The complex-step differentiation is a similar approach exploiting complex numbers arithmetics [84]. This method is more accurate since it avoids subtractive cancellation error as it does not involve a difference operation. Nevertheless, they both are brute force methods, requiring a profound number of arithmetic operations for each design and state variable. This family of methods is suitable for rough estimation of the outcome, or for "debugging" purposes, since few things can go wrong in its implementation.

Automatic differentiation represents much more sophisticated methodology to obtain the sensitivities of a given function, where one uses its programmatic representation to produce the analytic derivatives of a desired expression [68]. The key ingredient to this approach is that any function may be computed in a sequence of some elementary arithmetic operations, to which the chain rule of differentiation can be applied. It is possible to utilize this information to automatically generate a computer code producing the derivatives of a given function. Ultimately, model-based methods can be distinguished, which will be discussed in the following section.

2.4 Model-based sensitivity analysis

One of the most accurate and computationally efficient methods of calculating derivatives of performance measure and state variables with respect to design variables are based on the mathematical model of a system [85]. A multibody system is often formally modeled as a system of DAEs, which must be integrated numerically to get the state of the system in the next time instant. In the optimal control or design of MBS, an implicit dependency exists between state $\mathbf{y}$ and design variables $\mathbf{b}$, which further adds complexity to the problem. Two major model-based techniques exist in the literature for MBS sensitivity analysis: *direct differentiation method* and *adjoint variable method*.

Before discussing these topics in detail, it is worthwhile to introduce the notation behind the partial derivatives. Since the matrix calculus notation is employed throughout the thesis, the following abbreviation will be convenient. Suppose that $\mathbf{\Phi}(\mathbf{y}) \in \mathcal{R}^m$

denotes a column vector of m functions dependent on $\mathbf{y}$. The partial derivative of $\mathbf{\Phi}$ with respect to $\mathbf{y}$ can be shortly written as $\mathbf{\Phi}_\mathbf{y} \in \mathcal{R}^{m\times n}$ and is defined in the following way:

$$\mathbf{\Phi}_\mathbf{y} = \frac{\partial \mathbf{\Phi}}{\partial \mathbf{y}} = \begin{bmatrix} \frac{\partial \Phi_1}{\partial y_1} & \frac{\partial \Phi_1}{\partial y_2} & \dots & \frac{\partial \Phi_1}{\partial y_n} \\ \vdots & \ddots & \ddots & \vdots \\ \frac{\partial \Phi_m}{\partial y_1} & \dots & \dots & \frac{\partial \Phi_m}{\partial y_n} \end{bmatrix}. \tag{2.4}$$

If $\mathbf{\Phi}$ was a scalar function, the outcome of eq. (2.4) would produce a row vector (one of few matrix symbols in the text) rather than the more common column vector.

2.4.1 Direct differentiation method

The direct differentiation method (DDM) is conceptually simpler than the adjoint method. From a mathematical perspective, the DDM applies the chain rule of differentiation to the performance measure (for example, equation (2.1)), which can be written as:

$$\nabla_\mathbf{b} J = \int_0^{t_f} \left(h_\mathbf{y} \mathbf{y}_\mathbf{b} + h_{\dot{\mathbf{y}}} \dot{\mathbf{y}}_\mathbf{b} + h_\mathbf{b} \right) \mathrm{d}t + S_\mathbf{y} \mathbf{y}_\mathbf{b}|_{t_f} + S_{\dot{\mathbf{y}}} \dot{\mathbf{y}}_\mathbf{b}|_{t_f}. \tag{2.5}$$

The nabla operator ∇ refers to the gradient of the performance index J with respect to the variable appearing in the subscript. The key idea is to compute the implicit state derivatives $\mathbf{y}_\mathbf{b}, \dot{\mathbf{y}}_\mathbf{b}$ by differentiating the underlying dynamic equations (2.2) with respect to the design or control variables $\mathbf{b}$:

$$[\mathbf{F}_\mathbf{y}, \mathbf{F}_{\dot{\mathbf{y}}}, \mathbf{F}_\mathbf{v}] \begin{bmatrix} \mathbf{y}_\mathbf{b} \\ \dot{\mathbf{y}}_\mathbf{b} \\ \mathbf{v}_\mathbf{b} \end{bmatrix} = -\mathbf{F}_\mathbf{b}. \tag{2.6}$$

In the literature, equation (2.6) is referred to as a set of sensitivity equations [30]. In order to calculate sensitivity terms, eq. (2.6) must be numerically integrated along the solution of original equations of motion to get the derivative information. The initial condition for the sensitivity equations can be obtained by differentiating eq. (2.3) with respect to the vector of design variables. Consequently, inserting computed state derivatives into eq. (2.5) yields the gradient of the performance measure. Once obtained, these quantities can be used to calculate the gradient of each constraint equation [22, 36].

For a multibody system modeled by n generalized coordinates on which m algebraic constraints are imposed, and k design variables defined, a total number of $k \cdot (n + m)$ additional differential-algebraic equations must be integrated to calculate the sensitivity terms. Since the size of the sensitivity system is proportional to k, the direct approach

is applicable when the number of design variables is not prohibitively large, however, a threshold for which the method becomes computationally infeasible compared to the adjoint method seems to be rather difficult to grasp [74].

There are many advantages of the direct differentiation method. The method exhibits much higher numerical stability as compared to the finite difference method. The computational burden does not increase substantially when additional objective functions or constraints are added to the problem. Much research had been done in this field, which manifest itself in the wealth of approaches and implementations [18, 63, 87]. The direct method has been used in sensitivity analysis of flexible MBS [12, 34] as well as in problems involving nonholonomic constraints [37]. The author have also utilized the method in dynamic balancing of a four-bar linkage [78].

Concurrently, it is more often the case in the field of multibody dynamics that the number of constraints is relatively low compared with the dimensions of $\mathbf{b}$. The adjoint approach focuses on solving different equations, that avoids the cumbersome computation of the state derivatives. Therefore, the method's efficiency is not affected by the number of design/control variables, and the total cost of computing the gradient is proportional only to the size of the underlying dynamic problem. This property makes the adjoint method the most viable option for efficient and reliable gradient computation as will be shortly presented in the next subsection.

2.4.2 Adjoint method

The main idea of the adjoint method is to include the equality constraint equations (2.2) in the generic cost functional [53]. The extended performance measure (2.1) can be therefore written in the following way:

$$\overline{J} = \int_0^{t_f} \left[h(\mathbf{y}, \dot{\mathbf{y}}, \mathbf{b}) + \mathbf{w}^T \mathbf{F}(\mathbf{y}, \dot{\mathbf{y}}, \boldsymbol{\nu}, \mathbf{b}) \right] \, \mathrm{d}t + S(\mathbf{y}, \dot{\mathbf{y}})|_{t_f}, \tag{2.7}$$

where $\mathbf{w}(t) \in \mathcal{R}^n$ is a vector of arbitrary, time-dependent multipliers, known as *adjoint*, or *costate*, variables [4]. We can obtain a first-order description $\nabla_{\mathbf{b}} J$ of the effect of changing $\mathbf{b}$ on the magnitude of J by taking the first variation of Eq (2.7):

$$\begin{aligned} \delta\overline{J} = \int_0^{t_f} \big[& h_{\mathbf{b}} \delta\mathbf{b} + h_{\mathbf{y}} \delta\mathbf{y} + h_{\dot{\mathbf{y}}} \delta\dot{\mathbf{y}} + \mathbf{w}^T \mathbf{F}_{\mathbf{y}} \delta\mathbf{y} + \mathbf{w}^T \mathbf{F}_{\dot{\mathbf{y}}} \delta\dot{\mathbf{y}} \\ & + \mathbf{w}^T \mathbf{F}_{\mathbf{b}} \delta\mathbf{b} + \mathbf{w}^T \mathbf{F}_{\boldsymbol{\nu}} \delta\boldsymbol{\nu} \big] \, \mathrm{d}t + S_{\mathbf{y}} \delta\mathbf{y}|_{t_f} + S_{\dot{\mathbf{y}}} \delta\dot{\mathbf{y}}|_{t_f}. \end{aligned} \tag{2.8}$$

The symbol δ refers to the variation of a given expression [45]. The total variation of the performance measure $\delta\overline{J}$ is a sum of variations of its arguments, i.e., $\delta\mathbf{y}, \delta\boldsymbol{\nu}, \delta\mathbf{b}$,

premultiplied by the appropriate Jacobi matrices. The variation $\delta\dot{\mathbf{y}}$ that appears under the integral can be eliminated by integrating appropriate expression by parts:

$$\begin{aligned}\int_0^{t_f}\Big[\big(h_{\dot{\mathbf{y}}}+\mathbf{w}^T\mathbf{F}_{\dot{\mathbf{y}}}\big)\delta\dot{\mathbf{y}}\Big]\,\mathrm{d}t = -\int_0^{t_f}\Big[\Big(\frac{\mathrm{d}}{\mathrm{d}t}(h_{\dot{\mathbf{y}}})+\frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{w}^T\mathbf{F}_{\dot{\mathbf{y}}})\Big)\delta\mathbf{y}\Big]\,\mathrm{d}t \\ +\big(h_{\dot{\mathbf{y}}}|_{t_f}+\mathbf{w}^T\mathbf{F}_{\dot{\mathbf{y}}}\big)\delta\mathbf{y}|_{t_f}.\end{aligned} \tag{2.9}$$

Only final time component is taken into account outside of the integral in eq. (2.9) since $\delta\mathbf{y}|_{t=0}=\mathbf{0}$ when initial conditions are explicitly prescribed. Upon substituting equation (2.9) into eq. (2.8), we come up with the following expression:

$$\begin{aligned}\delta\overline{J} = \int_0^{t_f}\Big[h_{\mathbf{b}}\delta\mathbf{b}+\Big(h_{\mathbf{y}}-\frac{\mathrm{d}}{\mathrm{d}t}(h_{\dot{\mathbf{y}}})+\mathbf{w}^T\mathbf{F}_{\mathbf{y}}-\frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{w}^T\mathbf{F}_{\dot{\mathbf{y}}})\Big)\delta\mathbf{y}+\mathbf{w}^T\mathbf{F}_{\mathbf{b}}\delta\mathbf{b} \\ +\mathbf{w}^T\mathbf{F}_{\boldsymbol{\nu}}\delta\boldsymbol{\nu}\Big]\,\mathrm{d}t+\big(S_{\mathbf{y}}+h_{\dot{\mathbf{y}}}+\mathbf{w}^T\mathbf{F}_{\dot{\mathbf{y}}}\big)\delta\mathbf{y}|_{t_f}+\mathbf{S}_{\dot{\mathbf{y}}}\delta\dot{\mathbf{y}}|_{t_f}.\end{aligned} \tag{2.10}$$

Adjoint variables have at this point arbitrary values. The goal of the adjoint method is to select such a set of adjoint variables that would simplify equation (2.10) to the expression involving solely variations of design or control variables $\delta\mathbf{b}$:

$$\delta\overline{J}=\int_0^{t_f}\big(h_{\mathbf{b}}+\mathbf{w}^T\mathbf{F}_{\mathbf{b}}\big)\delta\mathbf{b}\,\mathrm{d}t, \tag{2.11}$$

Equation (2.11) exposes the gradient of the performance measure and allows us to obtain a unique expression for its computation by exploiting the adjoint variables:

- $\nabla_{\mathbf{b}}\overline{J}=\displaystyle\int_0^{t_f}\big(h_{\mathbf{b}}+\mathbf{w}^T\mathbf{F}_{\mathbf{b}}\big)\,\mathrm{d}t$ – when $\mathbf{b}$ represents design variables
- $\nabla_{\mathbf{b}}\overline{J}=(h_{\mathbf{b}}+\mathbf{w}^T\mathbf{F}_{\mathbf{b}})\Delta t$ – when $\mathbf{b}$ denotes discretized control input signals, and Δt is the discretization time-step.

To find a set of design variables or control functions that produces a stationary value of the cost functional J, the adjoint variables are required to fulfill the following necessary conditions:

$$\mathbf{F}_{\dot{\mathbf{y}}}^T\dot{\mathbf{w}}=h_{\mathbf{y}}^T-\dot{h}_{\dot{\mathbf{y}}}^T+(\mathbf{F}_{\mathbf{y}}^T-\dot{\mathbf{F}}_{\dot{\mathbf{y}}}^T)\mathbf{w}, \tag{2.12a}$$

$$\mathbf{F}_{\boldsymbol{\nu}}^T\mathbf{w}=\mathbf{0}, \tag{2.12b}$$

$$\mathbf{F}_{\dot{\mathbf{y}}}^T\mathbf{w}|_{t_f}=-S_{\mathbf{y}}^T-h_{\dot{\mathbf{y}}}^T|_{t_f}, \tag{2.12c}$$

$$S_{\dot{\mathbf{y}}}^T=\mathbf{0}, \tag{2.12d}$$

which is a general formula for the adjoint system. Equations (2.12a, 2.12b) constitute a set of DAEs that must be solved backward in time from the boundary conditions prescribed by eqs. (2.12c, 2.12d). Equation (2.12d) suggests that the terminal cost S does not depend on the time derivative $\dot{\mathbf{y}}$, which is against the assumption presented in eq. (2.7). This issue can be readily circumvented; however, it involves introducing additional components that would obscure the main point presented herein. The details on how one can treat the boundary conditions of the adjoint system are presented in section 4.2.

From the implementational point of view, the adjoint variable method consists of two major steps. For a given initial set of design variables, a forward dynamics problem (2.2), (2.3) is solved, whereas the state variables are recorded and reused later in the process [24]. Necessary conditions for an extremum of a cost functional (2.12) generate a system of differential-algebraic equations that must be solved backward in time in the second step of the method. Upon solving these equations, the adjoint variables can be utilized to evaluate the variation of performance measure in terms of variations of design variables via eq. (2.11). The cost involved in calculating sensitivities using the adjoint method is practically independent of the number of design variables k. In general, one would say that if the number of functions for which the sensitivity information must be obtained is smaller than the number of design variables k, then the adjoint method would be a better choice as compared to the direct differentiation method [8].

Frequently, the adjoint method is used in solving optimal control problems of multibody systems. The continuous problem might be reformulated then to a discrete setting by employing various numerical integration routines, such as trapezoidal rule [80] or HHT scheme [71]. Recently, the adjoint approach has been extended to cope with multibody systems involving discontinuous effects (impact or contact forces) [31, 103, 110] and bodies' flexibility [23, 58].

2.5 Thesis objectives

The presented case reveals a particular niche for an efficient, parallelizable framework based on indirect sensitivity analysis utilizing a multibody dynamics-specific structure. This thesis aims at responding to that particular demand by combining three major elements listed in fig. 2.2 and implementing them as a compound tool for a high-performant adjoint sensitivity analysis framework. The core idea behind the proposed method is to *combine the divide-and-conquer scheme with the indirect optimal control approach, specifically the adjoint method, in the multibody dynamics.* In light of the laid-out observations and examination of the state-of-the-art, we formulate the following thesis for this work:

Adjoint-based methods can be combined with constrained Hamiltonian dynamics formulations and parallel divide-and-conquer paradigm to provide efficient and reliable approach for optimal control or optimal design of high-degree-of-freedom multibody systems.

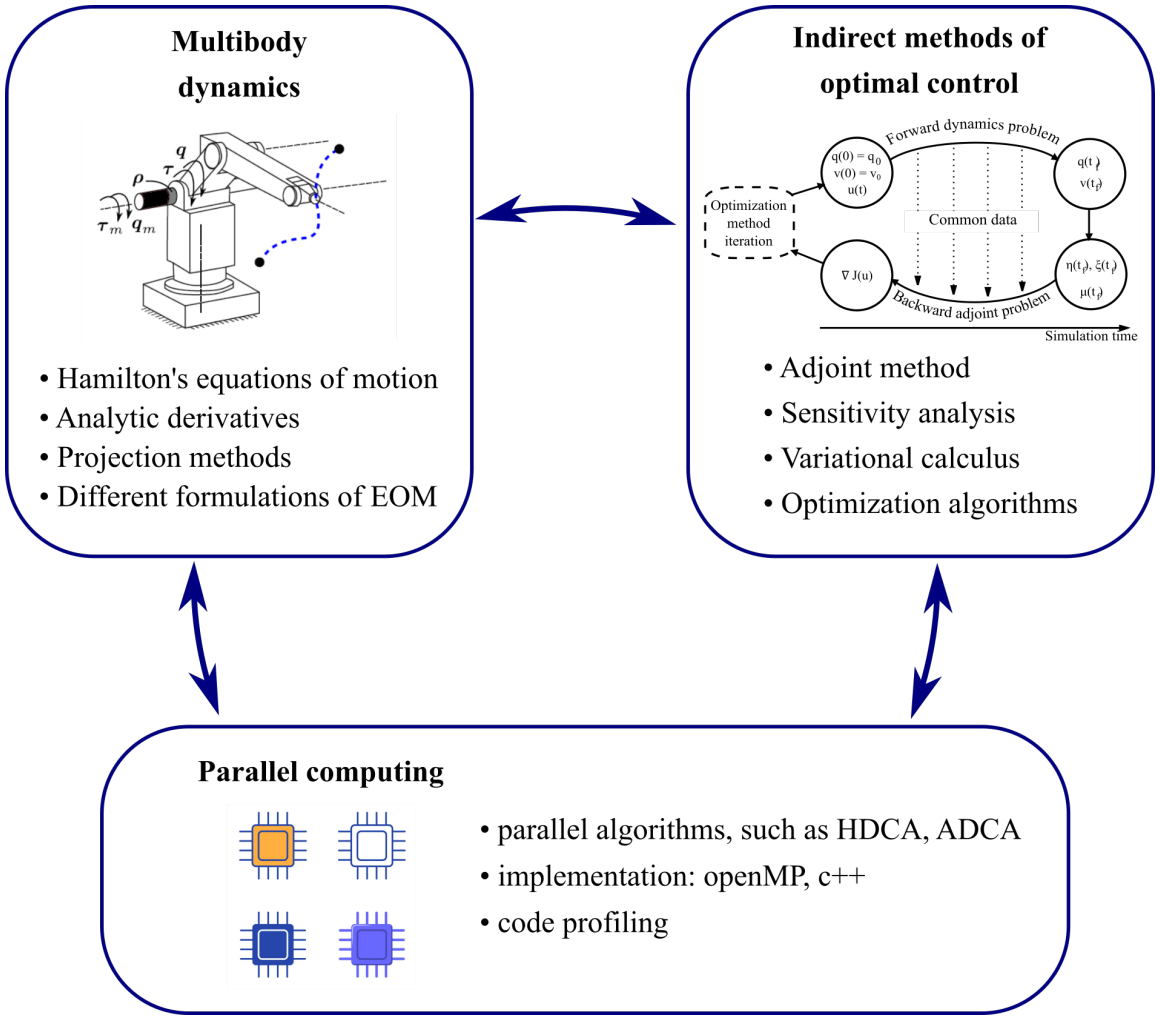


Figure 2.2: Scope of the thesis - a concise pictorial presentation

Fulfilling this task requires achieving multiple secondary goals that can be treated as additional contributions to this thesis. Let us list some significant scientific challenges (constituting scientific novelties in the field) that had to be addressed in the course of this work:

Goal 1 Development of the Hamiltonian-based adjoint method in absolute coordinates

We assume Hamilton's formalism as a primary tool for describing the MBS dynamics. According to the literature, the Hamiltonian approach is often more effi-

cient and numerically stable when compared to the acceleration-based formulation, mainly due to the lowered differential index of the underlying formulation [28]. As a result, algebraic constraints tend to degenerate over time at a much lower rate than acceleration-based counterparts, which relieves us from introducing artificial constraint stabilization quantities into the EOM (e.g., penalty terms [50]).

The coefficients of the adjoint system include partial derivatives of various quantities appearing in the underlying dynamic equations. One must calculate these expressions accurately to exploit the adjoint method reliably. The employed formulation of EOM (based on the angular velocity) poses additional complications for analyzing these quantities. The approach proposed in this thesis overcomes this issue and allows for reliable computation of these derivatives in an analytical fashion.

Goal 2 Development of the Hamiltonian-based adjoint method in mixed absolute-joint coordinates

It is a common practice in the field of multi-rigid body dynamics to employ a redundant set of coordinates. As a result, both the dynamic system of EOM and its adjoint are formulated as a set of differential-algebraic equations. This approach is attractive due to its relative simplicity; however, it leads to a mixed set of DAEs, which can be challenging to solve numerically.

On the other hand, the derivation of the adjoint system based on the joint-coordinate formulation of the underlying dynamics raises very complicated expressions for the coefficients of the resultant adjoint system. Although the outcome is a set of ODEs, the resulting quantities are much harder to establish systematically than in the case of the redundant DAE formulation [113]. Recent works investigate the relationships and analogies between the adjoint systems formulated as DAEs and ODEs [106], whereas joint-coordinate adjoint formulations are also actively studied in the literature [58].

Herein, a novel method is proposed that circumvents both highlighted problems. Initially formulated as a set of differential-algebraic equations, the adjoint system is brought into a minimal form by projecting the original expressions into the joint's motion and constraint force subspaces. This approach avoids computing cumbersome partial derivatives corresponding to joint-space equations. Furthermore, it establishes the relationships between redundant and independent adjoint variables. Ultimately, it allows for a derivation of the adjoint system formulated as a set of first-order ODEs with right-hand sides that can be easily presented in a closed form.

Goal 3 Development of the recursive, divide-and-conquer adjoint method

Little to no publications can be found regarding the crossover of the DCA, adjoint method, and multibody dynamics. This goal constitutes the main contribution of the thesis, which has been briefly summarized in figure 2.2. Integration of the adjoint method with a recursive algorithm based on the divide–and–conquer paradigm will be referred to as the Hamiltonian Adjoint-based Divide-and-Conquer Algorithm (HADCA).

Derivation of this approach is far from a straightforward task. Certain non-trivial relations and analogies between variables describing dynamic (physical) and adjoint (abstract) worlds must be established, e.g., spatial relations for adjoint variables along a rigid body. To the author's knowledge, these pieces of information are not available in the worldwide literature. Furthermore, the method depends on finding correct relationships between redundant and independent adjoint variables, as discussed in **Goal 2**. The overall shape of these newly developed equations will prove practical in exposing certain analogies between kinematic (tangible) and adjoint (abstract) variables.

Consequently, studied algorithms have been implemented as a code base allowing for testing the correctness of computations and benchmarking its performance. The results have been analyzed with respect to the problem size and the number of threads.Apart from scalability analysis, multiple logical sections of the algorithm have been minutely timed, and the results discussed.

Goal 4 Implementation of the developed methods in a simulation environment followed by validation of the results

This work presents multiple numerical examples ranging from academic benchmarks from literature to practical optimal control problems. The scope of each problem is unique in a specific manner. For example, certain examples address the problem of optimal control, while others pertain to optimal design. Considered multibody systems are characterized by diversified topologies: planar or spatial mechanisms with either closed- or open-loop kinematic chains. A number of problems involve the control of underactuated systems [107].

Numerical examples have been utilized to test the correctness and accuracy of the solutions proposed in this thesis. Furthermore, they allowed for testing the relationships established in the course of achieving the previous goals for the purpose of integrating the novel approach HADCA.

Goal 5 Implementation of the developed methods in the hardware

An electro-mechanical multibody model of the physical system was built, and its output was compared with the measured hardware response. The dynamic model involves additional relations describing the transmission and two electric motors. It was used to generate distinct trajectories of the end-effector along the manipulator's workspace. Consequently, the generated signals have been utilized in the control loop of the device as a feedforward signal. The developed model-based control algorithm was compared with a classical approach based on a PD controller and reported more accurate results with the former method. Last but not least, the proposed approach was used in a problem involving upper bounds on the input voltage, where a feasible trajectory was computed and executed on the hardware.

When formulating the objectives above, it was expected that the conducted research would lead to novel scientific results in the field of multibody dynamics, sensitivity analysis, and high-performance computing algorithms. The results of this study are discussed in the relevant sections of the dissertation and listed in the concluding chapter.

CHAPTER 3

MULTIBODY SYSTEM DYNAMICS IN THE HAMILTONIAN FRAMEWORK

3.1 Introduction

This chapter looks into the forward dynamics problem and delivers multiple approaches for formulating the equations of motion for multi-rigid-body systems. The common factor of all presented formulations is the fact, that they are based on the formulation of Hamilton's canonical equations. A global setting expressed by means of redundant, absolute coordinates has been presented in section 3.2. Absolute coordinates are highly suitable for the automatic generation of EOM since the resultant equations become much more straightforward compared to systems described by, e.g., joint coordinates. The Jacobian matrices in such systems are simpler to compute, and their structure is sparse. Joints are explicitly introduced by defining the kinematic constraints vector in the form of algebraic equations, which results in a dynamic system expressed as a set of DAEs. Although these procedures are relatively simple to formulate and solve, the numerical cost of calculations may be proportional up to N_b^3, where N_b is a number of bodies in the system.

Consequently, a more involved, yet highly efficient, formulation is presented in section 3.3, where equations of motion are generated recursively with the aid of the Hamiltonian Divide-and-Conquer algorithm (HDCA) [27, 28, 29]. This approach relies heavily on the concepts of spatial vectors [43], rigid body transformation matrices [64], and projection methods [20]. The primary variables for the integration procedure are, nevertheless, the joint-space coordinates, i.e., joint positions and joint momenta. In this approach, joints are modeled via two orthogonal subspaces $\mathbf{H}_i$ and $\mathbf{D}_i$, which uniquely define both allowable motion as well as constraint loads directions, respectively. By projecting the resulting equations onto these subspaces, the number of variables required to integrate

is equal to the number of degrees of freedom for an open-loop chain. Closed-loop chains require additional effort to get rid of cut-joint constraints from the formulation [28, 96]. Ultimately, we come up with a system of ordinary differential equations (ODEs), which can be solved more accurately and usually with lower effort than in the case of the system formulated globally.

3.2 Absolute coordinate formulation

The purpose of this section is to provide a reader with an overview of the forms of Hamilton's canonical equations of motion exploited in this thesis. The main focus is put on the global formulation of the equations of motion, where the primary variables are the absolute coordinates. The holonomic, bilateral, and scleronomic constraints are introduced in the text. Basic symbols that are used in the thesis are explained in detail.

3.2.1 Formulation based on the derivative of quaternions

The equations of motion can be described by means of various sets of coordinates [32, 55, 95, 124]. If the number of coordinates is greater than the number of the system's degrees of freedom, the algebraic constraints are introduced to express the relations between coordinates in a multibody system. Usually, these relations express the fact that bodies are connected by joints. Let us assume that m^* independent holonomic constraint equations, appended by N_b quaternion normalizing constraints, are represented as:

$$\mathbf{\Phi}(\mathbf{q}) = \mathbf{0}, \tag{3.1}$$

where $\mathbf{q} \in \mathcal{R}^{n_q}$ is a vector of n_q dependent coordinates, and $\mathbf{\Phi} \in \mathcal{R}^m$, $m = m^* + N_b$ is a vector of nonlinear constraint equations. Each entry of $\mathbf{q}$ includes a position vector ($\mathbf{r}_i \in \mathcal{R}^3$) and an orientation vector ($\mathbf{e} \in \mathcal{R}^4$) associated with i-th body, i.e.: $\mathbf{q}_i = [\mathbf{r}_i^T,\ \mathbf{e}_i^T]^T$ ($i = \{1, 2, \cdots N_b\}$). The orientation $\mathbf{e}$ is described by unit quaternions, also known as Euler parameters. These quantities must fulfill a normalization constraint $||\mathbf{e}||_2 = 1$, which introduces additional N_b mathematical constraint (in addition to the m^* geometric constraints) to the system (3.1). The time derivative of eq. (3.1) provides:

$$\dot{\mathbf{\Phi}}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{\Phi}_\mathbf{q}\dot{\mathbf{q}} = \mathbf{0}, \tag{3.2}$$

where $\mathbf{\Phi}_\mathbf{q} \in \mathcal{R}^{m \times n_q}$ denotes the constraints' Jacobian matrix with respect to coordinates $\mathbf{q}$. We would like to derive Hamilton's canonical equations which will exploit velocity level constraint equations (3.2). Let $\mathcal{L}$ be the system Lagrangian function defined by

$\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) = T(\mathbf{q}, \dot{\mathbf{q}}) - V(\mathbf{q})$, where $T = \frac{1}{2}\dot{\mathbf{q}}^T \mathbf{M}^{\#} \dot{\mathbf{q}}$ and V are the kinetic and potential energy of a system, respectively [9]. Symbol $\mathbf{M}^{\#} \in \mathcal{R}^{n_q \times n_q}$ denotes a mass matrix with the inertia component based on the derivative of quaternions. Let us also introduce a new unknown variable, canonical momenta $\mathbf{p}^{\#} \in \mathcal{R}^{n_q}$, conjugated with the velocities in the following way:

$$\mathbf{p}^{\#} = \left(\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}}\right)^T = \mathbf{M}^{\#} \dot{\mathbf{q}}, \tag{3.3}$$

Each sub-element of $\mathbf{p}^{\#}$ consists of (physical) 3×1 linear momenta vector and (abstract, bound with the derivative of the Euler parameters) 4×1 angular momenta. Assuming that eq. (3.3) holds, the Hamiltonian function of a multibody system can be expressed as $\mathcal{H} = {\mathbf{p}^{\#}}^T \dot{\mathbf{q}} - \mathcal{L}$. Hamilton's equations of motion for constrained multi-rigid-body system can be formulated as follows [32]:

$$\dot{\mathbf{q}} = \left(\frac{\partial \mathcal{H}}{\partial \mathbf{p}}\right)^T, \tag{3.4a}$$

$$\dot{\mathbf{p}}^{\#} = -\left(\frac{\partial \mathcal{H}}{\partial \mathbf{q}}\right)^T + \mathbf{f}_{ex}^{\#} - \mathbf{\Phi}_{\mathbf{q}}^T \boldsymbol{\lambda}, \tag{3.4b}$$

where $\mathbf{f}_{ex}^{\#} = \mathbf{f}_{ex}^{\#}(\mathbf{q}, \dot{\mathbf{q}}, t)$ is a vector of external and non-conservative forces, and the quantity $\boldsymbol{\lambda} \in \mathcal{R}^m$ denotes a vector of Lagrange multipliers that represent constraint forces at joints distributed along the directions indicated by columns of the Jacobian matrix $\mathbf{\Phi}_{\mathbf{q}}$. The kinetic energy of a MBS is a quadratic function of generalized velocities. Therefore, equations (3.4) can be transformed to the following form:

$$\mathbf{p}^{\#} = \mathbf{M}^{\#} \dot{\mathbf{q}}, \quad \dot{\mathbf{p}}^{\#} = \mathbf{f}^{\#} - {\mathbf{\Phi}_{\mathbf{q}}}^T \boldsymbol{\lambda}, \tag{3.5}$$

where $\mathbf{f}^{\#} = \mathbf{f}_{ex}^{\#} - \mathcal{H}_{\mathbf{q}}^T$ is a sum of external non-conservative and conservative forces, respectively. The equations of motion (3.5) together with the constraint equations (3.1) constitute a set of $2n_q + m$ differential-algebraic equations of index two [17] with position, momenta, and Lagrange multipliers as unknowns. According to the literature, this approach often proves to be more efficient and numerically stable, when compared to the acceleration-based formulation, mainly due to the lowered differential index of a DAE system [27, 17, 69, 16].

Now, let us augment the Lagrangian function with a term explicitly enforcing the kinematic velocity constraints (3.2): $\mathcal{L}^* = \mathcal{L} + \boldsymbol{\sigma}^T \dot{\mathbf{\Phi}}$. The quantity $\boldsymbol{\sigma} \in \mathcal{R}^m$ represents a vector of m new Lagrange multipliers associated with the velocity level constraint equations (3.2). Based on this modification, we can define the augmented momenta in the following way:

$$\mathbf{p}^{\#*} = \left(\frac{\partial \mathcal{L}^*}{\partial \dot{\mathbf{q}}}\right)^T = \mathbf{M}^{\#}\dot{\mathbf{q}} + \mathbf{\Phi}_{\mathbf{q}}^T \boldsymbol{\sigma}. \tag{3.6}$$

It can be easily shown that $\dot{\boldsymbol{\sigma}} = \boldsymbol{\lambda}$ [91]. The physical interpretation of this relation is that $\boldsymbol{\sigma}$ is a vector of impulses of constraint forces. Please note that the augmented momenta in eq. (3.6) are expressed as a sum of canonical momenta vector $\mathbf{p}$ and impulses of constraint loads distributed along constrained directions indicated by the Jacobian matrix. Moreover, the derivative of eq. (3.6) can be written in two forms. Directly, it reads:

$$\dot{\mathbf{p}}^{\#*} = \mathbf{M}^{\#}\ddot{\mathbf{q}} + \dot{\mathbf{M}}^{\#}\dot{\mathbf{q}} + \mathbf{\Phi}_{\mathbf{q}}^T \boldsymbol{\lambda} + \dot{\mathbf{\Phi}}_{\mathbf{q}}^T \boldsymbol{\sigma}. \tag{3.7}$$

Subsequently, by using Eqs. (3.5) and the property $\dot{\boldsymbol{\sigma}} = \boldsymbol{\lambda}$, we would come up with the following formula for the time derivative of the augmented momentum vector:

$$\dot{\mathbf{p}}^{\#*} = \mathbf{f}^{\#} + \dot{\mathbf{\Phi}}_{\mathbf{q}}^T \boldsymbol{\sigma}. \tag{3.8}$$

The fact that $\boldsymbol{\lambda}$ is the time derivative of the vector $\boldsymbol{\sigma}$ signifies important consequences. Firstly, an initial value of $\boldsymbol{\sigma}$ can be chosen arbitrarily, e.g. $\boldsymbol{\sigma}(0) = \mathbf{0}$. In some situations, its value may rapidly grow for long-time simulations, especially when a particular reaction force doesn't change sign over certain time interval. In order to overcome this potential difficulty, one may reset the values of constraint impulsive forces by setting them to zero once they exceed a certain threshold and recalculate the values of $\mathbf{p}^{\#*}$ and $\dot{\mathbf{p}}^{\#*}$ in accordance with Eqs. (3.6) and (3.8). Let us emphasize, that such an artificial discontinuity of constraint impulse loads does not affect the forward dynamics solutions since only the time derivative of $\boldsymbol{\sigma}$ has a tangible physical interpretation. Nevertheless, an artificial discontinuity in $\boldsymbol{\sigma}$ might have an impact on the way the adjoint method is formulated and thus on the resultant adjoint equations. Recent works of Serban [110], Corner [31], and Pikuliński [103] indicate that it is possible to introduce discontinuous behavior into the adjoint sensitivity formulation, however, in further derivations it is assumed that $\boldsymbol{\sigma}$ is a continuous-time function.

The solution of the EOM associated with augmented momenta can be realized in two steps. Given the initial conditions for the system in the form $\mathbf{q}(0) = \mathbf{q}^{(0)}$ and $\mathbf{p}^{\#*}(0) = \mathbf{p}^{\#*(0)}$ in the first step[1], a linear system of equations (3.2), (3.6) is solved with respect to velocity $\dot{\mathbf{q}}$ and Lagrange multipliers $\boldsymbol{\sigma}$. The solution of these equations gives the necessary unknown inputs to the second step of the procedure. Subsequently, vector $\dot{\mathbf{p}}^{\#*}$ is evaluated directly from eq. (3.8). The time derivatives $\begin{bmatrix}\dot{\mathbf{q}}_i^T & \dot{\mathbf{p}}_i^{\#*T}\end{bmatrix}^T$ are numerically integrated to

[1] Assuming that $\boldsymbol{\sigma}^{(0)} = \mathbf{0}$, initial value of the augmented momenta vector can be simply computed as $\mathbf{p}^{\#(0)} = \mathbf{M}^{\#}(\mathbf{q}^{(0)}) \cdot \dot{\mathbf{q}}^{(0)}$.

obtain the state of the system in the next time instant. Equations (3.2, 3.6, 3.8) can be gathered to yield the following set of equations of motion:

$$\begin{bmatrix} \mathbf{M}^{\#}(\mathbf{q}) & \mathbf{\Phi}_{\mathbf{q}}^{T}(\mathbf{q}) \\ \mathbf{\Phi}_{\mathbf{q}}(\mathbf{q}) & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}} \\ \boldsymbol{\sigma} \end{bmatrix} = \begin{bmatrix} \mathbf{p}^{\#*} \\ \mathbf{0} \end{bmatrix} \tag{3.9}$$
$$\dot{\mathbf{p}}^{\#*} = \mathbf{f}^{\#}(\mathbf{q}, \dot{\mathbf{q}}) + \dot{\mathbf{\Phi}}_{\mathbf{q}}^{T}(\mathbf{q}, \dot{\mathbf{q}})\boldsymbol{\sigma}.$$

Let us note, that this is the equivalent of the general, implicit equation (2.2) with the following substitutions: $\mathbf{y} := [\mathbf{q}^T, \mathbf{p}^{\#*T}]^T$ and $\boldsymbol{\nu} := \boldsymbol{\sigma}$.

Furthermore, one can calculate accelerations and reaction forces with small additional computational cost since both mass and constraint Jacobian matrices necessary to get $\mathbf{p}^{\#*}$ and $\dot{\mathbf{p}}^{\#*}$ have already been factorized and stored in a computer memory. The time derivative of eq. (3.2) yields a vector of acceleration-level constraints:

$$\ddot{\mathbf{\Phi}} = \mathbf{\Phi}_{\mathbf{q}}\dot{\mathbf{q}} - \boldsymbol{\gamma}^{\#} = \mathbf{0}, \quad \boldsymbol{\gamma}^{\#} = -\dot{\mathbf{\Phi}}_{\mathbf{q}}\dot{\mathbf{q}}, \tag{3.10}$$

which can be combined with eq. (3.7) to obtain the value of $\ddot{\mathbf{q}}$ and $\boldsymbol{\lambda}$. Let us note that certain constraint violation errors are expected at the position level, since the position-level constraints are not explicitly enforced. Nevertheless, the position constraint violation errors tend to increase only linearly with respect to time for the Hamiltonian-based formulation expressed as index-2 DAEs. The errors at the position level for basic acceleration based formulations (formulated as index-1 DAEs) have a quadratic trend. Although certain formulations manage to fulfill constraints at all levels [54, 16], it should be pointed out that the Hamiltonian based formulation used herein neither defines local parametrizations of the constraint manifold to reduce the number of coordinates to the minimum set nor it introduces artificial constraint stabilization quantities into the EOM (such as e.g. penalty terms [50]).

3.2.2 Formulation based on spatial velocity

A spatial velocity vector associated with i-th body, i.e., $\mathbf{v}_i = [\dot{\mathbf{r}}_i^T, \boldsymbol{\omega}_i^{(i)T}]^T \in \mathcal{R}^6$, is related to the derivative $\dot{\mathbf{q}}_i$ via a bidirectional, configuration dependent, map [55]. Globally, it can be written as:

$$\dot{\mathbf{q}} = \mathbf{L}_h^T(\mathbf{q})\mathbf{v}, \qquad \mathbf{v} = \mathbf{L}_d(\mathbf{q})\dot{\mathbf{q}} \tag{3.11}$$

where $\mathbf{L}_d$ and $\mathbf{L}_h^T$ are the following block-diagonal matrices:

$$\mathbf{L}_d = \begin{bmatrix} \ddots & & \\ & \mathbf{1}_{3\times3} \quad \mathbf{0} & \\ & \mathbf{0} \quad 2\,\mathbf{L}_i & \\ & (\boldsymbol{double}\nearrow) & \ddots \end{bmatrix}, \qquad \mathbf{L}_h^T = \begin{bmatrix} \ddots & & \\ & \mathbf{1}_{3\times3} \quad \mathbf{0} & \\ & \mathbf{0} \quad 0.5\,\mathbf{L}_i^T & \\ & (\boldsymbol{half}\nearrow) & \ddots \end{bmatrix}, \tag{3.12}$$

and $\mathbf{L}_i \in \mathcal{R}^{3\times4}$ is a transformation matrix of the Euler parameters' derivatives onto the angular velocity vector [56]. The superscript (i) signifies that $\boldsymbol{\omega}_i^{(i)} \in \mathcal{R}^3$ is expressed in the reference frame of i^{th} body. Furthermore, it is a vector quantity, which is not the case for $\dot{\mathbf{e}}_i$. Due to this fact, the use of the spatial velocity vector $\mathbf{v}$ as a set of six-dimensional generalized velocities leads to much simpler equations of motion. We can enforce this fact, by inserting equation (3.11) into eq. (3.2):

$$\dot{\mathbf{\Phi}}(\mathbf{q}, \mathbf{v}) = \mathbf{\Phi}_\mathbf{q} \cdot \mathbf{L}_h^T \mathbf{v} = \mathbf{C}\mathbf{v} = \mathbf{0}. \tag{3.13}$$

The Jacobian $\mathbf{\Phi}_\mathbf{q}$ is simply a partial derivative of eq. (3.1) with respect to vector $\mathbf{q}$, which – as long as the rotational components are concerned – is not the case for $\mathbf{C}$. The latter quantity can be obtained by calculating a *virtual rotation* [55] of the constraints vector (3.1) (c.f. appendix A).

A virtual rotation is an infinitesimal rotation applied to the body with time held fixed. It can be shown, that virtual rotation (contrary to a regular rotation) is in fact a vector quantity; however, similarly to the angular velocity, it is non-integrable. Thus, during the integration of the equations of motion, the time derivatives of the generalized coordinates $\dot{\mathbf{q}}$ must be extracted with the aid of eq. (3.11). The concept of virtual rotations and differentiability of the velocity will be significant in the following chapter. Example A.1 presented in appendix A aims to bring these topics closer by presenting a simple derivation of a quantity that arises in the dynamic equations.

Ultimately, the equivalent of equation (3.9) based on angular velocity has the following form:

$$\begin{bmatrix} \mathbf{M}(\mathbf{q}) & \mathbf{C}^T(\mathbf{q}) \\ \mathbf{C}(\mathbf{q}) & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \boldsymbol{\sigma} \end{bmatrix} = \begin{bmatrix} \mathbf{p}^* \\ \mathbf{0} \end{bmatrix}, \tag{3.14a}$$

$$\dot{\mathbf{p}}^* = \mathbf{f}(\mathbf{q}, \mathbf{v}) + \dot{\mathbf{C}}^T(\mathbf{q}, \mathbf{v})\boldsymbol{\sigma}. \tag{3.14b}$$

The mass matrix $\mathbf{M} \in \mathcal{R}^{n_v \times n_v}$ is defined for the $i-th$ body as $\mathbf{M}^i = \text{diag}(m_i \cdot \mathbf{I}, \mathbf{J}_i)$, where m_i is the mass, and $\mathbf{J}_i \in \mathcal{R}^{3\times3}$ denotes the inertia tensor of the body. Furthermore, the

coefficient matrices in both formulations are closely related, i.e., $\mathbf{M}^{\#} = \mathbf{L}_d^T \mathbf{M} \mathbf{L}_d$ and $\mathbf{f} = \mathbf{L}_h \mathbf{f}^{\#} = \mathbf{f}_{ex} - \mathcal{H}_{\boldsymbol{\pi}'}^T$, where $\mathcal{H}_{\boldsymbol{\pi}'}^T$ denotes a partial derivative of the Hamiltonian in the sense of virtual displacements and virtual rotations (i.e., $\mathcal{H}_{\boldsymbol{\pi}'}^T$ constitutes coefficients standing next to the virtual quantities after taking a variation of $\mathcal{H}$). Moreover, augmented momenta $\mathbf{p}^* \in \mathcal{R}^{n_v}$ are tied with absolute velocity vector $\mathbf{v}$ instead of $\dot{\mathbf{q}}$, which was the case in eq. (3.9). Similarly, equation (3.10) can be rewritten in the following way: $\ddot{\boldsymbol{\Phi}} = \mathbf{C}\dot{\mathbf{v}} - \boldsymbol{\gamma}$, where $\boldsymbol{\gamma} = -\mathbf{C}\mathbf{v}$, which yields acceleration-level equations for $\dot{\mathbf{v}}$ an reaction forces $\boldsymbol{\lambda}$:

$$\begin{bmatrix} \mathbf{M} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{v}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{f} - \dot{\mathbf{M}}\mathbf{v} \\ \boldsymbol{\gamma} \end{bmatrix}, \tag{3.15}$$

A closer look at equations (3.14a), (3.15) reveals that the mathematical constraints imposed on the Euler parameters vanish since stacked 6-vector of spatial velocities is the unknown variable instead of a stacked 7-vector $\dot{\mathbf{q}}$. This has an important consequence: the matrix $\mathbf{M}^{\#}$ is always singular [2], whereas the matrix $\mathbf{M}$ is invertible in non-singular configurations. In the following part of this chapter, this requirement will be required for deriving EOM based on the divide-and-conquer paradigm. Consequently, this property renders the equations (3.14) as much more useful in further derivations. Therefore, we need to rely on the mathematical apparatus associated with virtual rotations to obtain state derivatives during the sensitivity analysis. The details of this calculus are presented in the appendix A.

Finally, let us elaborate on the notation that will be utilized in the remainder of this thesis. One can see that the hash symbol (#) is used to distinguish between different formulations presented in this section. Nevertheless, both equation (3.9) and eq. (3.14) are formulated globally, where all coordinates are stacked into one global vector for a given quantity. The following sections will introduce a joint-space formulation of EOM for which it is more convenient to consider the dynamic equations with respect to a single body rather than globally. Subsequently, the distinction between global and local (body-wise, i.e., referring to a single body) formulation is emphasized by the case: lower case refers to the global formulation, whereas upper case distinguishes local formulation. Table 3.1 presents a summary of some symbols that appear throughout this work. It includes parameters that will be defined in the following section for completeness. Last but not least, all plotted quantities will be expressed in SI units unless stated otherwise.

[2] It is possible update a rank of the mass matrix (e.g., for the purpose of obtaining the Hamiltonian via Legendre transform) by substituting $\mathbf{M}_4^{\#\text{non-singular}} := \mathbf{M}_4^{\#\text{singular}} + 2\text{Tr}(M)\mathbf{e}\mathbf{e}^T$. Nevertheless, the physical interpretation of the variables associated with the mass matrix $\mathbf{M}$ (i.e. velocities $\mathbf{v}$ and momenta $\mathbf{p}^*$) is much clearer when compared with variables corresponding to $\mathbf{M}^{\#}$.

Table 3.1: Symbols used throughout the thesis to describe quantities in different formulations (abbr. "form.")

quantity:	symbol for:			
	stacked 7×1 form.	stacked 6×1 form.	body-wise 6×1 form.	joint space
velocity	$\dot{\mathbf{q}}$	$\mathbf{v}$	$\mathbf{V}$	$\boldsymbol{\beta}$
generalized force	$\mathbf{f}^{\#}$	$\mathbf{f}$	$\mathbf{F}$	n.a.
phys. momenta	$\mathbf{p}^{\#}$	$\mathbf{p}$	$\mathbf{P}$	$\widehat{\mathbf{p}}$
aug. momenta	$\mathbf{p}^{\#*}$	$\mathbf{p}^{*}$	n.a.	$\widehat{\mathbf{p}}$

3.3 Joint coordinate formulation

The main objective of this section is to present the recursive Hamiltonian Divide-and-Conquer algorithm for forward dynamics problem. Joint coordinate formulation of the MBS is introduced and employed throughout the text. We begin by explaining basic terminology and defining useful spatial operators. Following that, recursive relations for DCA scheme are established and presented in a fine detail.

3.3.1 Joint-space coordinates, spatial operators, and motion subspaces

The terminology associated with the topology of an open-loop multibody systems is quite straightforward and universal throughout the literature [91, 64, 43]. Let us consider a MBS presented in figure 3.1. A base (or ground) body, denoted by identification number (id) 0, is assumed to be stationary. The first body attached to the base is referred to as the *root* body (id = 1). It can be treated as a *child* of the ground body as well a *parent* for the subsequent (id=2) body. Bodies may possess multiple topological children (e.g. body i and its children j, k); however, we will assume that each body (excluding the base-body) can have only one parent. It means that the topology of the considered MBS is tree-structured. All children bodies appearing along a kinematic chain constitute a subset of *descendants* of a given body.

Since the number of joints is equal to the number of bodies[3], we conveniently assign an identification number of each child body to the id of the corresponding joint. For example, a kinematic joint j couples two neighboring bodies: parent body i and a child body j. The notation presented in the text should capture an important distinction that arises between bodies and joints. To this end, two arbitrary, neighboring links, specifically bodies i and j, are distinguished and given labels A and B, respectively (c.f. fig. 3.1).

[3]Here, we implicitly assumed an open kinematic chain. In closed kinematic chains, joints related to loop-closing constraints can have a separate id number.

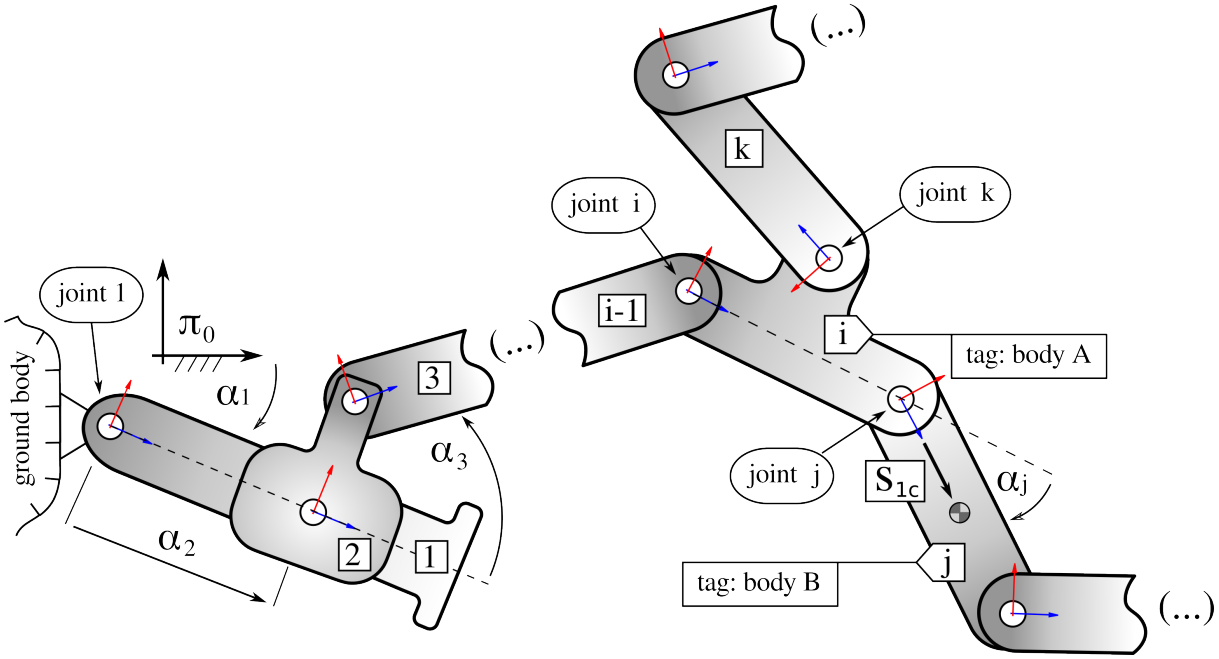


Figure 3.1: Multibody system with joints and labeling scheme

When referring to various quantities, such as velocities, momenta, or subspaces, we will hold to the convention, where symbols i and j refer to the joints, whereas characters A and B apply to the bodies. Nevertheless, it is correct to treat these symbols as interchangeable, e.g. $\widehat{\mathbf{p}}_i = \widehat{\mathbf{p}}_A, \boldsymbol{\sigma}_j = \boldsymbol{\sigma}_B$, etc. (c.f. eq. (3.22)).

Suppose that a multi-rigid-body system consists of N_b bodies and possesses N^{dof} degrees of freedom. The configuration state of a system can be described with the aid of joint coordinates $\boldsymbol{\alpha} \in \mathcal{R}^{n_\alpha}, n_\alpha \geqslant N^{dof}$, and the velocity state space by means of joint velocities $\boldsymbol{\beta} \in \mathcal{R}^{n_\beta}, n_\beta = N^{dof}$. Similarly to their absolute coordinate counterparts (c.f. equation (3.11)), the derivative of joint coordinates is related to joint velocities via bidirectional, configuration-dependent maps ${}^{\dot{\alpha}}\mathbf{B}_{\beta} \in \mathcal{R}^{n_\alpha \times n_\beta}$ and ${}^{\beta}\mathbf{B}_{\dot{\alpha}} \in \mathcal{R}^{n_\beta \times n_\alpha}$:

$$\dot{\boldsymbol{\alpha}} = {}^{\dot{\alpha}}\mathbf{B}_{\beta}\boldsymbol{\beta}, \qquad \boldsymbol{\beta} = {}^{\beta}\mathbf{B}_{\dot{\alpha}}\dot{\boldsymbol{\alpha}} \tag{3.16}$$

We denote the absolute orientation of i-th body with the matrix $\mathbf{A}_i \in \mathcal{R}^{3\times 3}$, whereas the relative rotation between two adjacent bodies is expressed by $\mathbf{A}^{(i,j)}$. Hence, the following relation is true:

$$\mathbf{A}(\boldsymbol{\alpha}_j) = \mathbf{A}^{(i,j)} = \mathbf{A}_i^T \cdot \mathbf{A}_j. \tag{3.17}$$

Let us highlight a particular point (or points) on a body, called *handle*, by means of which the body can interact with the rest of a MBS via both active and constraint force components. A joint coupling two bodies can thus be substituted by two handles; one on each body. When a body has two handles, we will often need to express its absolute

spatial velocity at one handle with respect to the velocity of its other handle. This relation can be conveniently written for an arbitrary body A with the aid of the shift matrix [43], i.e.:

$$\mathbf{V}_2^A = \begin{bmatrix} \dot{\mathbf{r}}_2^{A^{(0)}} \\ \boldsymbol{\omega}_2^{A^{(0)}} \end{bmatrix} = \left(\mathbf{S}_{12}^A\right)^T \mathbf{V}_1^A. \tag{3.18}$$

Here $\mathbf{V}_1^A$ and $\mathbf{V}_2^A$ are, respectively, the absolute spatial velocity of body A at its first and second handle, and $\mathbf{s}_{12}$ denotes the vector connecting these handles, whereas:

$$\mathbf{S}_{12}^A = \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \widetilde{\mathbf{s}}_{12}^{(0)} & \mathbf{I}_{3\times3} \end{bmatrix} \quad \text{and} \quad \widetilde{\mathbf{s}} = \begin{bmatrix} 0 & -s_z & s_y \\ s_z & 0 & -s_x \\ -s_y & s_x & 0 \end{bmatrix}. \tag{3.19}$$

In this section, both linear and angular components of the spatial velocity $\mathbf{V}$ are represented in inertial reference frame ($(i) = (0)$). [4] For brevity, we omit the symbol indicating frame 0 when this is the case.

Each kinematic pair can be defined by a *joint's motion subspace* $\mathbf{H}_i(\boldsymbol{\alpha}_i) \in \mathcal{R}^{6\times N^{\mathrm{dof}_i}}$ [43] ($N_{dof} \in \langle 1; 5\rangle$ is a number of the joint's degrees of freedom), which maps joint velocities onto the spatial velocity vectors, i.e.:

$$\Delta\mathbf{V}_j = \mathbf{H}_j(\boldsymbol{\alpha}_j) \cdot \boldsymbol{\beta}_j. \tag{3.20}$$

The spatial velocity in eq. (3.20) is a relative velocity transferred through the j-th joint, here represented in the global coordinate frame. Some particular subspaces, together with the corresponding constrained velocity subspaces, are presented in table 3.2. In order to express the absolute spatial velocity of a point on the particular body, a recursive relation must be established:

$$\mathbf{V}_1^B = \left(\mathbf{S}_{12}^A\right)^T \mathbf{V}_1^A + \mathbf{H}_j \boldsymbol{\beta}_j \tag{3.21}$$

[4] The angular velocity in sec. 3.2 was represented in a body reference frame. When comparing spatial velocities of two bodies, their angular velocities must be adjusted; hence, if this were the case, equation (3.21) would become:

$$\mathbf{V}_1^B = \boldsymbol{\gamma}^{(B,A)}\left(\left(\mathbf{S}_{12}^A\right)^T \mathbf{V}_1^A + \mathbf{H}_j^{(A)} \boldsymbol{\beta}_j\right) = \left({}^{\gamma}\mathbf{S}_{12}^A\right)^T \mathbf{V}_1^A + \mathbf{H}_j^{(B)} \boldsymbol{\beta}_j,$$

$$\left({}^{\gamma}\mathbf{S}_{12}^A\right)^T = \boldsymbol{\gamma}^{(B,A)} \left(\mathbf{S}_{12}^A\right)^T = \begin{bmatrix} \mathbf{1}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{A}^{(B,A)} \end{bmatrix} \left(\mathbf{S}_{12}^A\right)^T.$$

The term $\boldsymbol{\gamma}^{(B,A)}$ must be introduced when the angular velocity vector is expressed in a local coordinate system of a different body. This notation clearly hinders the readability and adds little valuable insight. Therefore, we describe the angular velocity in a global reference frame throughout section (3.3) (c.f. eq. (3.18)).

The recursion begins at the root body ($j = 1$), for which we can apply equation (3.20): $\mathbf{V}_1 = \mathbf{H}_1 \cdot \boldsymbol{\beta}_1$. A graphical representation of eq. (3.21) for the case of a revolute joint is visualized in figure 3.2a.

Table 3.2: Orthogonal subspaces $\mathbf{H}$ and $\mathbf{D}$ defining kinematic properties of certain joints represented in the parent body reference frame

spherical joint		revolute joint (about x axis)		universal joint, $\boldsymbol{\alpha} \in \mathcal{R}^2$ (cross defined by axes x and y)	
$\mathbf{H}$	$\mathbf{D}$	$\mathbf{H}$	$\mathbf{D}$	$\mathbf{H}(\boldsymbol{\alpha})$	$\mathbf{D}(\boldsymbol{\alpha})$
$\begin{bmatrix} \mathbf{0}_{3\times3} \\ \mathbf{I}_{3\times3} \end{bmatrix}$	$\begin{bmatrix} \mathbf{I}_{3\times3} \\ \mathbf{0}_{3\times3} \end{bmatrix}$	$\begin{bmatrix} \mathbf{0}_{3\times1} \\ 1 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} \mathbf{I}_{3\times1} \\ 0 \\ 1 \\ 1 \end{bmatrix}$	$\begin{bmatrix} \mathbf{0}_{3\times2} \\ 1 \quad 0 \\ 0 \quad \cos\alpha_1 \\ 0 \quad \sin\alpha_1 \end{bmatrix}$	$\begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times1} \\ & 1 \\ \mathbf{0}_{3\times3} & -\sin\alpha_1 \\ & \cos\alpha_1 \end{bmatrix}$

Relation (3.3) implies that a joint velocity transferred via allowable motion subspace must have an equivalent joint momentum, which will be referred to as $\widehat{\mathbf{p}} \in \mathcal{R}^{n_\beta}$. Moreover, by finding a null space of $\mathbf{H}_i$, one can define a *constrained velocity subspace* $\mathbf{D}_i(\boldsymbol{\alpha}_i) \in \mathcal{R}^{6\times(6-N_i^{\text{dof}})}$ [64]. Both subspaces are orthogonal to each other, thus: $\mathbf{D}_i^T\mathbf{H}_i = \mathbf{0}$. Moreover, it is always possible to define $\mathbf{H}_i$ and $\mathbf{D}_i$ for a single joint so that the following conditions are met: $\mathbf{H}_i^T\mathbf{H}_i = \mathbf{I}$, $\mathbf{D}_i^T\mathbf{D}_i = \mathbf{I}$, where the symbol $\mathbf{I}$ denotes a unity matrix of appropriate size. Altogether, these quantities can be utilized to describe active and constraint force components emerging in a joint. Specifically, a *spatial articulated momentum* vector $\overline{\mathbf{P}}_1^A \in \mathcal{R}^6$ of linear and angular momenta related to the first handle of the body A is defined as:

$$\overline{\mathbf{P}}_1^A = \mathbf{H}_i\widehat{\mathbf{p}}_i + \mathbf{D}_i\boldsymbol{\sigma}_i, \tag{3.22}$$

where $\boldsymbol{\sigma}_i$ denotes impulses of constraint loads associated with the i-th kinematic joint. The term *articulated* refers to this quantity being the local (i.e., associated with a given handle) distribution of the translational and angular momenta. In other words, the physical momentum vector of a body's geometrical point can be expressed as a sum of all articulated momenta defined for the body. As it was the case with spatial velocities, we will need a way to represent an articulated momentum vector at one point of the body (e.g. handle 2 of body A) with respect to the other point (e.g. handle 1). Once again, shift matrix defined in eq. (3.19) turns out to be convenient utility for this task:

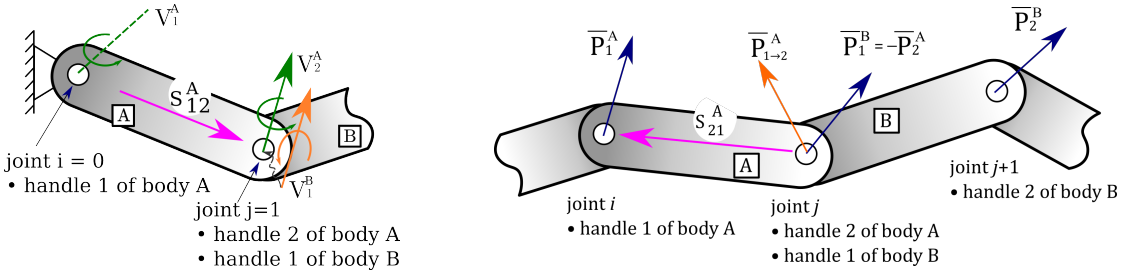
$$\overline{\mathbf{P}}_{1\to2}^A = \mathbf{S}_{21}^A\overline{\mathbf{P}}_1^A \tag{3.23}$$

The same transformation holds for spatial momenta as well as spatial force vectors (c.f. fig. 3.2b). Moreover, shift matrix can be defined as a global matrix that acts between

stacked vectors, which allows to rewrite eq. (3.23) with quantities defined in section 3.2, i.e. $\mathbf{p}^*$, $\dot{\mathbf{p}}^*$. It is important not to confuse two different quantities, which are $\overline{\mathbf{P}}^A_{1\to 2}$ of eq. (3.23) and $\overline{\mathbf{P}}^A_2$ given by:

$$\overline{\mathbf{P}}^A_2 = -\boldsymbol{\gamma}^{(A,B)}\overline{\mathbf{P}}^B_1 = -\overline{\mathbf{P}}^B_1. \tag{3.24}$$

The former denotes the articulated momentum vector of joint i shifted along body A to its second handle, whereas the latter is an articulated momentum associated with j-th joint represented in the coordinate frame of body A.



(a) Velocity transformation

(b) Articulated momenta transformation

Figure 3.2: Different transformations of spatial vectors along a rigid body

3.3.2 Joint-space EOM in global formulation

This subsection presents the derivation of joint-space Hamilton's equations expressed as a global system of ODEs. The key feature of this calculation is based on the orthogonality of allowable and constrained motion subspaces. Equation (3.18) relates absolute velocity with joint velocity of a single body; however, stacking absolute velocities into a global vector, combined with eq. (3.20), yields the global expression: $\mathbf{v} = \mathbf{H}\boldsymbol{\beta}$. Matrix $\mathbf{H} \in \mathcal{R}^{n_v \times n_\beta}$ (with the subindices dropped) is a globally defined motion subspace. Since $\boldsymbol{\beta}$ is a vector of unconstrained variables for open-chain systems, the following relation also holds (c.f. eq. (3.13)): $\mathbf{CH} = \mathbf{0}$. Note that this is equivalent to the relation $\mathbf{D}_i^T\mathbf{H}_i = \mathbf{0}$ from previous subsection. A projection of the first set of equations in eq. (3.14a) onto the subspace $\mathbf{H}$ leads to the following equation:

$$\mathbf{H}^T\mathbf{p}^* = \mathbf{H}^T\mathbf{M}\mathbf{H}\boldsymbol{\beta} \quad \Rightarrow \quad \widehat{\mathbf{p}} = \widehat{\mathbf{M}}\boldsymbol{\beta}, \tag{3.25}$$

where $\widehat{\mathbf{p}} = \mathbf{H}^T\mathbf{p}^*$ denotes joint momenta in stacked vector format, whereas $\widehat{\mathbf{M}} = \mathbf{H}^T\mathbf{M}\mathbf{H}$ can be interpreted as joint-space mass matrix. Consequently, the derivative of $\widehat{\mathbf{p}}$ can be written as: $\dot{\widehat{\mathbf{p}}} = \mathbf{H}^T\dot{\mathbf{p}}^* + \dot{\mathbf{H}}^T\mathbf{p}^*$. Now, let us premultiply eq. (3.14b) with $\mathbf{H}^T$ and add

the term $\dot{\mathbf{H}}^T\mathbf{p}^*$ to both sides of the resultant equation. This allows for the following derivation:

$$\begin{aligned}\mathbf{H}^T\dot{\mathbf{p}}^* + \dot{\mathbf{H}}^T\mathbf{p}^* &= \mathbf{H}^T\mathbf{f} + \mathbf{H}^T\dot{\mathbf{C}}^T\boldsymbol{\sigma} + \dot{\mathbf{H}}^T\mathbf{p}^* \\ \dot{\widehat{\mathbf{p}}} &= \mathbf{H}^T\mathbf{f} + \mathbf{H}^T\dot{\mathbf{C}}^T\boldsymbol{\sigma} + \dot{\mathbf{H}}^T(\mathbf{M}\mathbf{v} + \mathbf{C}^T\boldsymbol{\sigma}) \\ &= \mathbf{H}^T\mathbf{f} + \dot{\mathbf{H}}^T\mathbf{M}\mathbf{v} + (\mathbf{H}^T\dot{\mathbf{C}}^T + \dot{\mathbf{H}}^T\mathbf{C}^T)\boldsymbol{\sigma}.\end{aligned} \tag{3.26}$$

By recalling that $\mathbf{H}^T\dot{\mathbf{C}}^T + \dot{\mathbf{H}}^T\mathbf{C}^T = \frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{H}^T\mathbf{C}^T) = \mathbf{0}$, we come up with the following set of equations:

$$\widehat{\mathbf{p}} = \widehat{\mathbf{M}}\boldsymbol{\beta}, \tag{3.27a}$$

$$\dot{\widehat{\mathbf{p}}} = \mathbf{H}^T\mathbf{f} + \dot{\mathbf{H}}^T\mathbf{M}\mathbf{H}\boldsymbol{\beta}. \tag{3.27b}$$

Equations (3.27) are applicable only to open kinematic chains of MBS. Additional m_c loop-closing constraint equations $\boldsymbol{\Theta} = \mathbf{0}, \boldsymbol{\Theta} \in \mathcal{R}^{m_c}$ must be defined in the case of closed-loop topology. Specifically, the kinematic chain is virtually cut at a selected joint to obtain an open-loop system [96]. The derivative of the constraints can be written as: $\dot{\boldsymbol{\Theta}} = \boldsymbol{\Gamma}\boldsymbol{\beta} = \mathbf{0}$, where $\boldsymbol{\Gamma} \in \mathcal{R}^{m_c \times n_\beta}$ is a Jacobian matrix that corresponds to loop-closure constraints. By following the steps presented in section 3.2.1 (c.f. eqs.(3.5) and (3.9)), we can include the specified constraints into EOM in the following way:

$$\widehat{\mathbf{p}} = \widehat{\mathbf{M}}\boldsymbol{\beta} + \boldsymbol{\Gamma}\boldsymbol{\sigma}^c, \tag{3.28a}$$

$$\dot{\widehat{\mathbf{p}}} = \mathbf{H}^T\mathbf{f} + \dot{\mathbf{H}}^T\mathbf{M}\mathbf{H}\boldsymbol{\beta} + \dot{\boldsymbol{\Gamma}}\boldsymbol{\sigma}^c, \tag{3.28b}$$

where $\boldsymbol{\sigma}^c \in \mathcal{R}^{m_c}$ defines impulses of constraint loads associated with loop-closure constraints.

3.4 The Divide-and-Conquer Algorithm

When solving forward dynamics problem, we can utilize one of two main approaches [43]:

- Form an equation of motion for the whole system, and solve it for the reaction variables and either momenta derivatives or (in classical formulations) accelerations
- Propagate constraints from one body to the next in such a way that the unknown variables can be calculated one joint at a time.

Sections 3.2 and 3.3 presented methods that fall into the former category. Methods of the second category work by calculating the coefficients associated with dynamic equations

that unknown variables must satisfy locally. Subsequently, these coefficients are *propagated* along the kinematic chain until it is possible to solve the dynamics locally at one particular – abstract, non-tangible – body (encompassing the entire dynamical relation of the MBS). Once the solution for this abstract body is known, we can recursively solve neighboring bodies until the whole problem is resolved.

The divide-and-conquer algorithm (DCA) is a member of the propagation family of methods, and the remainder of this chapter will be devoted to its description. The (DCA) is a parallelizable algorithm for calculating the forward dynamics of a general multi-rigid-body system. It can be applied to mechanisms with any topology or joint type, including branches and kinematic loops (herein, we adhere to the earlier assumption of tree-structured topology). The algorithm is non-iterative; however, it requires at least two recursions along the kinematic tree of the MBS. It features $O\log(N_b)$ time complexity on $O(N_b)$ processors and is the fastest available algorithm for a computer with a large number of processors and low communications costs [42].

The DCA works by recursively applying a formula that constructs the articulated body equations of motion of an assembly from its constituent parts. On the fundamental level, the algorithm is supplied with two independent subassemblies and a description specifying the type of connection in between. Subsequently, it yields the EOM of the assembly that can be interpreted as a single (articulated) body whose internal dynamics is a "black box". In the Hamiltonian setting, this is mathematically equivalent to dislodging the unknown reaction impulses by substituting them with an explicit expression and, in turn, generating the coefficients of the equations of motion for the whole assembly.

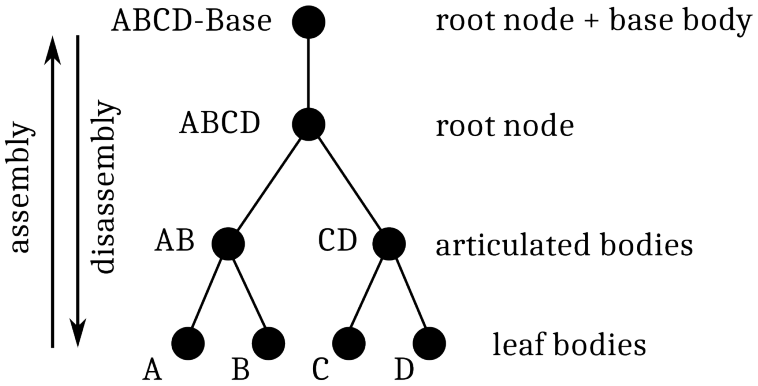


Figure 3.3: Binary tree representing recursive assembly-disassembly of a four-link multibody system

The pattern in which the recursion is applied over the kinematic chain can be represented by a binary tree (c.f. fig 3.3), where the (rigid, physical) leaf-bodies constitute its bottom nodes, and the root node represents a single compound body encompassing the

whole MBS. The nodes in-between are called articulated or compound bodies, and they represent abstract subset of the multibody system. Starting at the leaf-body level, the algorithm progresses along the tree until the root node's equations of motion have been constructed. One can solve the articulated system at this node by invoking connection conditions between root node and the base body. Subsequently, the algorithm traverses the binary tree backward until the leaf nodes have been solved, yielding the unknown variables, such as velocities or derivatives of joint momenta. This process is visualized in figure 3.4.

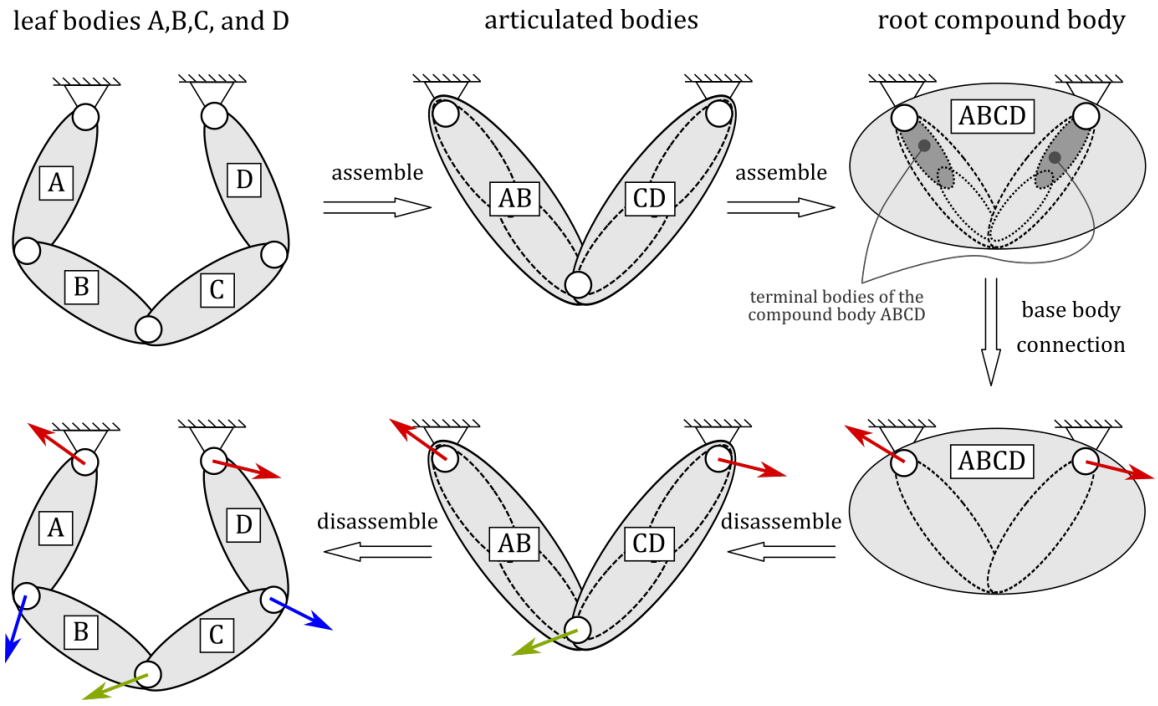


Figure 3.4: Graphical visualization of the divide-and-conquer procedure

The primary variables of the algorithm are joint coordinates. The algorithm utilizes both joint-space and absolute coordinates, where the former set is numerically integrated. Therefore, prior to the assembly/disassembly process one needs to extract absolute variables from their joint-space counterparts via recursive relations. Absolute velocity can be obtained with the aid of relation (3.21), whereas generalized coordinates must be evaluated via non-linear recursive relation, that can be symbolically expressed as $\mathbf{q} = \mathbf{g}(\boldsymbol{\alpha})$.

3.4.1 Divide-and-Conquer Algorithm: velocity-level equations

The goal of this subsection is to establish relations allowing for the computation of unknown joint variables, namely $\dot{\boldsymbol{\alpha}}$ and $\hat{\dot{\mathbf{p}}}$, in a recursive, divide-and-conquer manner. To

this end, we will utilize articulated momenta introduced in section 3.3.1 to derive the first set of recursive Hamiltonian dynamic equations of motion, equivalent to eq. (3.4a) [28]. Joint coordinates and momenta are treated as known parameters, provided by initial conditions or integration routine.

The physical linear and angular momentum $\mathbf{P} = \mathbf{MV}$ can be expressed as a sum of all articulated momenta distributed over a given body Let us assume that the MBS consists of a sequential chain of bodies, and each body has one child only. Consequently, any arbitrary body has two handles (associated with a parent- and a child-body). The case with a higher number of connections can be easily generalized [43]. The physical momentum vector evaluated at handle 1 may be represented as:

$$\begin{aligned}\mathbf{P}_1^A = \mathbf{M}_1^A \mathbf{V}_1^A &= \overline{\mathbf{P}}_1^A + \mathbf{S}_{12}^A \overline{\mathbf{P}}_2^A \\ &= \mathbf{H}_i \widehat{\mathbf{p}}_i + \mathbf{T}_1^A - \mathbf{S}_{12}^A (\mathbf{H}_j \widehat{\mathbf{p}}_j + \mathbf{T}_1^B),\end{aligned} \tag{3.29}$$

where symbols $\mathbf{T}_1^A = \mathbf{D}_i \boldsymbol{\sigma}_i$, $\mathbf{T}_1^B = \mathbf{D}_j \boldsymbol{\sigma}_j$ denote impulses of constraint loads acting on joints i (handle 1 of body A) and j (handle 1 of body B), respectively. Let us now consider two sets of articulated subassemblies A and B connected by a specific joint, e.g. revolute or translational joint. Each subassembly holds two terminal bodies, indicated by subscripts 1 and 2. An example of the terminal bodies for an articulated system consisting of four moving physical bodies has been presented in fig. 3.4. Please note that these bodies must not be directly attached to the ground body. For example, the terminal bodies of subassembly AB are bodies A and B. Based on eq. (3.29), velocities of each handle can be expressed as follows (c.f. eq. (4.49) in example 4.1):

$$\mathbf{V}_1^A = \boldsymbol{\zeta}_{11}^A \mathbf{T}_1^A + \boldsymbol{\zeta}_{12}^A \mathbf{T}_2^A + \boldsymbol{\zeta}_{10}^A, \tag{3.30}$$

$$\mathbf{V}_2^A = \boldsymbol{\zeta}_{21}^A \mathbf{T}_1^A + \boldsymbol{\zeta}_{22}^A \mathbf{T}_2^A + \boldsymbol{\zeta}_{20}^A, \tag{3.31}$$

$$\mathbf{V}_1^B = \boldsymbol{\zeta}_{11}^B \mathbf{T}_1^B + \boldsymbol{\zeta}_{12}^B \mathbf{T}_2^B + \boldsymbol{\zeta}_{10}^B, \tag{3.32}$$

$$\mathbf{V}_2^B = \boldsymbol{\zeta}_{21}^B \mathbf{T}_1^B + \boldsymbol{\zeta}_{22}^B \mathbf{T}_2^B + \boldsymbol{\zeta}_{20}^B. \tag{3.33}$$

Here, the unknown variables are impulses of constraint loads $\mathbf{T}_1^A, \mathbf{T}_2^A, \mathbf{T}_1^B, \mathbf{T}_2^B \in \mathcal{R}^6$, whereas symbols denoted by $\boldsymbol{\zeta}_{ij} \in \mathcal{R}^{6\times 6}$ and $\boldsymbol{\zeta}_{i0} \in \mathcal{R}^6$ represent known coefficients of the EOM (please refer to eq. (4.49) for a detailed description). The goal is to construct a more general articulated set C from subassemblies A and B, which can be expressed in the following way:

$$\mathbf{V}_1^C = \boldsymbol{\zeta}_{11}^C \mathbf{T}_1^C + \boldsymbol{\zeta}_{12}^C \mathbf{T}_2^C + \boldsymbol{\zeta}_{10}^C, \tag{3.34}$$

$$\mathbf{V}_2^C = \boldsymbol{\zeta}_{21}^C \mathbf{T}_1^C + \boldsymbol{\zeta}_{22}^C \mathbf{T}_2^C + \boldsymbol{\zeta}_{20}^C. \tag{3.35}$$

To this end, we consider equation (3.21), which relates velocities at the point of connection of the two subassemblies, and project it onto the subspace $\mathbf{D}_j$:

$$\mathbf{D}_j^T (\mathbf{V}_1^B - \mathbf{V}_2^A) = \mathbf{0}. \tag{3.36}$$

The orthogonality property $\mathbf{D}_j^T \cdot \mathbf{H}_j = \mathbf{0}$ and eq. (3.18) are used to simplify the expression. According to equation (3.24), impulses of constraint loads acting between the subassemblies are equal to $\mathbf{T}_1^B = -\mathbf{T}_2^A = \mathbf{D}_j \boldsymbol{\sigma}_j$. We can eliminate these unknown variables by substituting eqs. (3.31) and (3.32) into eq. (3.36):

$$\mathbf{D}_j^T (\boldsymbol{\zeta}_{11}^B \mathbf{D}_j \boldsymbol{\sigma}_j + \boldsymbol{\zeta}_{12}^B \mathbf{T}_2^B + \boldsymbol{\zeta}_{10}^B - \boldsymbol{\zeta}_{21}^A \mathbf{T}_1^A + \boldsymbol{\zeta}_{22}^A \mathbf{D}_j \boldsymbol{\sigma}_j - \boldsymbol{\zeta}_{20}^A) = \mathbf{0}. \tag{3.37}$$

Subsequently, equation 3.37 may be rewritten into the following form:

$$\mathbf{B}_j \boldsymbol{\sigma}_j = \underbrace{\Big[- \mathbf{D}_j^T (\boldsymbol{\zeta}_{11}^B + \boldsymbol{\zeta}_{22}^A) \mathbf{D}_j \Big]}_{\mathbf{B}_j} \boldsymbol{\sigma}_j = \mathbf{D}_j^T (-\boldsymbol{\zeta}_{21}^A \mathbf{T}_1^A + \boldsymbol{\zeta}_{12}^B \mathbf{T}_2^B + \boldsymbol{\zeta}_{10}^B - \boldsymbol{\zeta}_{20}^A) \tag{3.38}$$

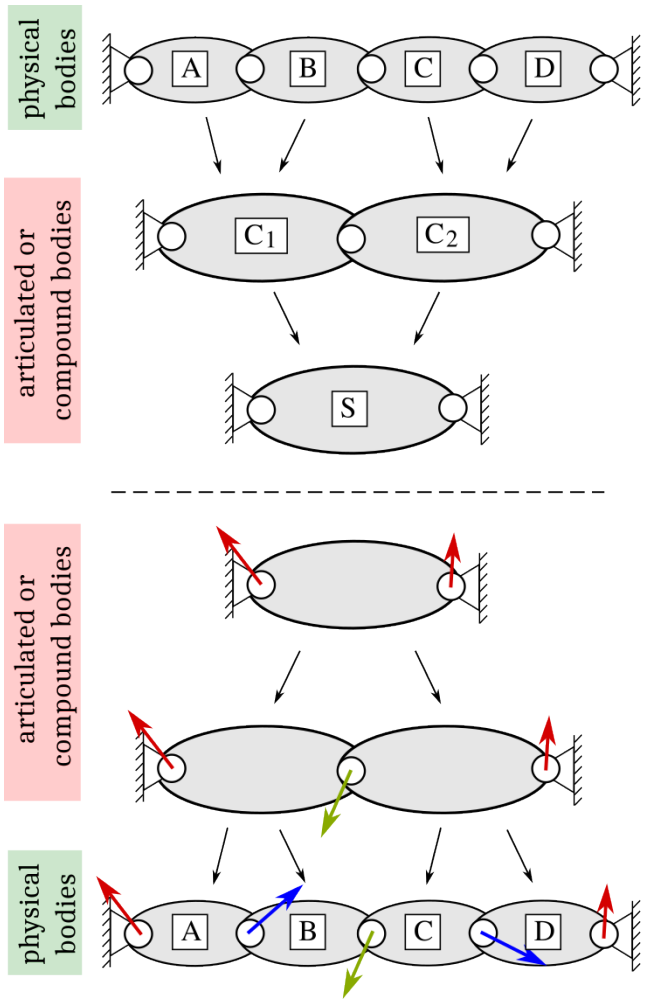
The matrices $\boldsymbol{\zeta}_{11}^B$ and $\boldsymbol{\zeta}_{22}^A$ are symmetric and, as long as no redundant constraints appear in the MBS, positive definite, while $\mathbf{D}_j$ has full rank. Thus, matrix $\mathbf{B}_j$ is non-singular and invertible. Consequently, $\boldsymbol{\sigma}_j$ can be explicitly extracted from eq. (3.38) by inverting matrix $\mathbf{B}_j$. First, let us define auxiliary symbols $\mathbf{W}_j = \mathbf{D}_j \mathbf{B}_j^{-1} \mathbf{D}_j^T$, $\mathbf{c}_j = \mathbf{W}_j (\boldsymbol{\zeta}_{10}^B - \boldsymbol{\zeta}_{20}^A)$ and premultiply equation (3.38) by the product $\mathbf{D}_j \mathbf{B}_j^{-1}$:

$$\begin{aligned} \mathbf{D}_j \boldsymbol{\sigma}_j &= \mathbf{W}_j (-\boldsymbol{\zeta}_{21}^A \mathbf{T}_1^A + \boldsymbol{\zeta}_{12}^B \mathbf{T}_2^B) + \mathbf{c}_j \\ &= \mathbf{T}_1^B = -\mathbf{T}_2^A \end{aligned} \tag{3.39}$$

Let us remark that eq. (3.39) will provide an important role in the following part of the algorithm. Now, we can substitute it back into eqs. (3.30), (3.33) to yield:

$$\mathbf{V}_1^A = +\Big(\boldsymbol{\zeta}_{11}^A + \boldsymbol{\zeta}_{12}^A \mathbf{W}_j \boldsymbol{\zeta}_{21}^A\Big) \mathbf{T}_1^A - \Big(\boldsymbol{\zeta}_{12}^A \mathbf{W}_j \boldsymbol{\zeta}_{12}^B\Big) \mathbf{T}_2^B + \Big(\boldsymbol{\zeta}_{10}^A - \boldsymbol{\zeta}_{12}^A \mathbf{c}_j\Big). \tag{3.40}$$

$$\mathbf{V}_2^B = -\Big(\boldsymbol{\zeta}_{21}^B \mathbf{W}_j \boldsymbol{\zeta}_{21}^A\Big) \mathbf{T}_1^A + \Big(\boldsymbol{\zeta}_2^B + \boldsymbol{\zeta}_{21}^B \mathbf{W}_j \boldsymbol{\zeta}_{12}^B\Big) \mathbf{T}_2^B + \Big(\boldsymbol{\zeta}_{20}^B + \boldsymbol{\zeta}_{21}^B \mathbf{c}_j\Big). \tag{3.41}$$



Coefficients $\boldsymbol{\zeta}$ are obtained from kinematic relations between physical (leaf) bodies.

Coefficients $\boldsymbol{\zeta}$ become recursively-evaluated abstract quantities; however, kinematic relations between articulated bodies are fulfilled

The orthogonality of subspaces **H** and **D** allow us to compute reaction components for the root compound body.

Once outermost reaction loads are known, it is possible to compute intermediate terms via disassembly phase.

Spatial velocities have been expressed w.r.t. reaction loads. Extracting these terms for all physical bodies allow us to calculate their spatial velocities.

Figure 3.5: Stages of the DCA procedure at the velocity level

By comparing equations (3.40), (3.41) with eqs. (3.34), (3.35) we come up with the following formulas for a divide-and-conquer generation of the compound bodies:

$$\begin{aligned}
\boldsymbol{\zeta}_{11}^{C} &= \boldsymbol{\zeta}_{11}^{A} + \boldsymbol{\zeta}_{12}^{A}\mathbf{W}_j\boldsymbol{\zeta}_{21}^{A}, & \boldsymbol{\zeta}_{12}^{C} &= -\boldsymbol{\zeta}_{12}^{A}\mathbf{W}_j\boldsymbol{\zeta}_{12}^{B}, \\
\boldsymbol{\zeta}_{21}^{C} &= -\boldsymbol{\zeta}_{21}^{B}\mathbf{W}_j\boldsymbol{\zeta}_{21}^{A}, & \boldsymbol{\zeta}_{22}^{C} &= \boldsymbol{\zeta}_{22}^{B} + \boldsymbol{\zeta}_{21}^{B}\mathbf{W}_j\boldsymbol{\zeta}_{12}^{B}, \\
\boldsymbol{\zeta}_{10}^{C} &= \boldsymbol{\zeta}_{10}^{A} - \boldsymbol{\zeta}_{12}^{A}\mathbf{c}_j, & \boldsymbol{\zeta}_{20}^{C} &= \boldsymbol{\zeta}_{20}^{B} + \boldsymbol{\zeta}_{21}^{B}\mathbf{c}_j,
\end{aligned} \tag{3.42}$$

At this point, we are able to express the unknown impulses of constraint loads acting between two articulated bodies in terms of known coefficients of the equations of motion. One may say that bodies A and B are "assembled" into body C. Hence, this part of the algorithm is referred to as the *assembly phase*. Equations (3.42) are applied recursively to each pair of articulated bodies starting at the leaf-body level where coefficients $\boldsymbol{\zeta}$ are determined with the aid of eq. (3.29). The process follows along a hierarchic binary tree defined by the particular topology of the MBS. Ultimately, a top node (labeled as $\mathcal{S}$) of

the tree is reached, which involves one large compound body comprising an entire system of bodies. In the most general case, the velocities of its external handles have the following structure:

$$\mathbf{V}_1^{\mathcal{S}} = \boldsymbol{\zeta}_{11}^{\mathcal{S}}\mathbf{T}_1^{\mathcal{S}} + \boldsymbol{\zeta}_{12}^{\mathcal{S}}\mathbf{T}_2^{\mathcal{S}} + \boldsymbol{\zeta}_{10}^{\mathcal{S}}, \tag{3.43a}$$

$$\mathbf{V}_2^{\mathcal{S}} = \boldsymbol{\zeta}_{21}^{\mathcal{S}}\mathbf{T}_1^{\mathcal{S}} + \boldsymbol{\zeta}_{22}^{\mathcal{S}}\mathbf{T}_2^{\mathcal{S}} + \boldsymbol{\zeta}_{20}^{\mathcal{S}}. \tag{3.43b}$$

We can highlight three boundary cases for connection of the compound body with a stationary ground body:

1. No connection at all – the multibody system is a free-floating object, such as a drone or a space manipulator. Both terminal reaction impulses are equal to zero: $\mathbf{T}_1^{\mathcal{S}} = \mathbf{0}, \mathbf{T}_2^{\mathcal{S}} = \mathbf{0}$.

2. The mechanism has an open kinematic loop, i.e. a tree-like structure. This case will be investigated more thoroughly.

3. The articulated compound body is connected to the ground body at two or more of its handles. The former case is equivalent to the equation (3.43).

The cases above are exhaustively explained in refs. [28, 42]. For simplicity, herein, we focus on the case of an open kinematic loop where $\mathbf{T}_2^{\mathcal{S}} = \mathbf{0}$. From the coincidence of the handles between compound body $\mathcal{S}$ and the physical body A, we can deduce that $\mathbf{V}_1^{\mathcal{S}} = \mathbf{H}_1\boldsymbol{\beta}_1$ and $\mathbf{T}_1^{\mathcal{S}} = \mathbf{D}_1\boldsymbol{\sigma}_1$. Plugging these relations into eq. (3.43a) followed by a projection of this equation onto the subspace $\mathbf{D}_1$ results in the relation for boundary impulses of constraint loads:

$$\left(\mathbf{D}_1^T\boldsymbol{\zeta}_{11}^{\mathcal{S}}\mathbf{D}_1\right)\boldsymbol{\sigma}_1 = -\mathbf{D}_1^T\boldsymbol{\zeta}_{10}^{\mathcal{S}}. \tag{3.44}$$

Solving equation (3.44) for $\boldsymbol{\sigma}_1$ paves the way for the next stage of the algorithm, labeled as *disassembly phase*. At this point, both terminal reaction impulses, i.e., $\mathbf{T}_1^{\mathcal{S}}$ and $\mathbf{T}_2^{\mathcal{S}}$, are known quantities. The compound body $\mathcal{S}$ can now be treated as a set of two articulated subassemblies A and B. The coincidence of handles provides that $\mathbf{T}_1^A = \mathbf{T}_1^{\mathcal{S}}$ as well as $\mathbf{T}_2^B = \mathbf{T}_2^{\mathcal{S}}$. It is possible to evaluate intermediate reaction loads $\mathbf{T}_1^B$ (and hence $\mathbf{T}_2^A$) by invoking equation (3.39). The compound body $\mathcal{S}$ has therefore been *disassembled* into two smaller subassemblies. This process is repeated from the root node downwards the binary tree until leaf nodes are reached. One can see that it is highly amenable to parallelization. As a result, all impulses of constraint loads are obtained, which in turn

allows calculating all spatial velocities by recalling eqs. (3.30)-(3.33). Furthermore, joint velocities can be computed by projecting eq. (3.21) onto the subspace $\mathbf{H}_j$:

$$\boldsymbol{\beta}_j = \mathbf{H}_j^T \left(\mathbf{V}_1^B - \mathbf{V}_2^A \right). \tag{3.45}$$

Ultimately, relation (3.16) is employed to map the stacked vector of joint velocities $\boldsymbol{\beta}$ into the derivative of joint coordinates vector, i.e. $\dot{\boldsymbol{\alpha}}$. The stages of a divide-and-conquer procedure, supplied with a brief summary of each step, have been visualized in figure 3.5.

3.4.2 Divide-and-Conquer Algorithm: force-level equations

The procedure presented in subsection 3.4.1 allows for a recursive solution of eq. (3.4a) in a parallel manner. A similar scheme may be proposed to compute the remaining equation (3.4b). The divide-and-conquer approach can be utilized again to express the distribution of active forces at each handle of all bodies in the MBS. The goal is, therefore, to decompose active force $\mathbf{Q}_1^A$ (expressed at handle 1) of the arbitrary body A into the following sum:

$$\mathbf{Q}_1^A = \overline{\mathbf{Q}}_1^A + \mathbf{S}_{12}^A \overline{\mathbf{Q}}_2^A, \tag{3.46}$$

where $\overline{\mathbf{Q}}_1^A$ and $\overline{\mathbf{Q}}_2^A$ are *articulated forces* acting on handles 1 and 2 of the body A. To this end, a recursive procedure can be applied that consists of two stages: assembly of the *accumulated forces* and their disassembly into the articulated forces. Accumulated force $\mathbf{Q}_1^A$ is the equivalent generalized force reduced for the whole articulated body. Generalized force has been introduced in sec. 3.2.1: $\mathbf{F}_1^A = \left[\mathbf{f}_{ex} - \mathcal{H}_{\pi'}^T \right] \big|_{bodyA}$ (c.f. eq. (3.14) and tab. 3.1). At the leaf-body level these quantities are equivalent to one another.

The assembly phase requires considering two neighboring subassemblies A and B:

$$\mathbf{Q}_1^A = \overline{\mathbf{Q}}_1^A + \mathbf{S}_{12}^A \overline{\mathbf{Q}}_2^A, \tag{3.47a}$$

$$\mathbf{Q}_1^B = \overline{\mathbf{Q}}_1^B + \mathbf{S}_{12}^B \overline{\mathbf{Q}}_2^B \tag{3.47b}$$

and constructing a compound assembly: $\mathbf{Q}_1^C = \overline{\mathbf{Q}}_1^C + \mathbf{S}_{12}^C \overline{\mathbf{Q}}_2^C$. Consequently, the following relationships hold: $\overline{\mathbf{Q}}_1^C = \overline{\mathbf{Q}}_1^A$, $\overline{\mathbf{Q}}_2^C = \overline{\mathbf{Q}}_2^B$ (the handles' coincidency condition), and $\overline{\mathbf{Q}}_2^A = -\overline{\mathbf{Q}}_1^B$ (the law of action-reaction). By inserting the last relation into eq. (3.47a), premultiplying eq. (3.47b) with $\mathbf{S}_{12}^A$, and summing both equations, we come up with the following formula:

$$\mathbf{Q}_1^A + \mathbf{S}_{12}^A \cdot \mathbf{Q}_1^B = \overline{\mathbf{Q}}_1^A + \mathbf{S}_{12}^A \mathbf{S}_{12}^B \overline{\mathbf{Q}}_2^B, \tag{3.48}$$

which suggests that

$$\mathbf{Q}_1^C = \mathbf{Q}_1^A + \mathbf{S}_{12}^A \cdot \mathbf{Q}_1^B. \tag{3.49}$$

Please note, that the operator $\mathbf{S}_{12}^A$ in eq. (3.48) refers to the whole subassembly A and may describe translation across multiple physical bodies. As in the previous part, the assembly phase begins at the leaf-body level, where force quantities are known. If the value of force does not depend explicitly on the velocity, it is possible to evaluate equation (3.49) in parallel with eqs. (3.42).

At some point, a root node of the binary tree is reached, which can be represented as $\mathbf{Q}_1^{\mathcal{S}}$. By invoking the base body connection conditions, we can deduce that: $\overline{\mathbf{Q}}_1^{\mathcal{S}} = \mathbf{Q}_1^{\mathcal{S}}$ and $\overline{\mathbf{Q}}_2^{\mathcal{S}} = \mathbf{0}$. The latter relation is also true in the case of closed kinematic chains, since only active components are taken into account. Once the terminal articulated forces are obtained, it is possible to evaluate intermediate quantities by transforming eq. (3.47b): $\overline{\mathbf{Q}}_1^B = \mathbf{Q}_1^B - \mathbf{S}_{12}^B \overline{\mathbf{Q}}_2^{\mathcal{S}} = \mathbf{Q}_1^B$. This procedure can be repeated recursively along all branches of the binary tree in the following manner:

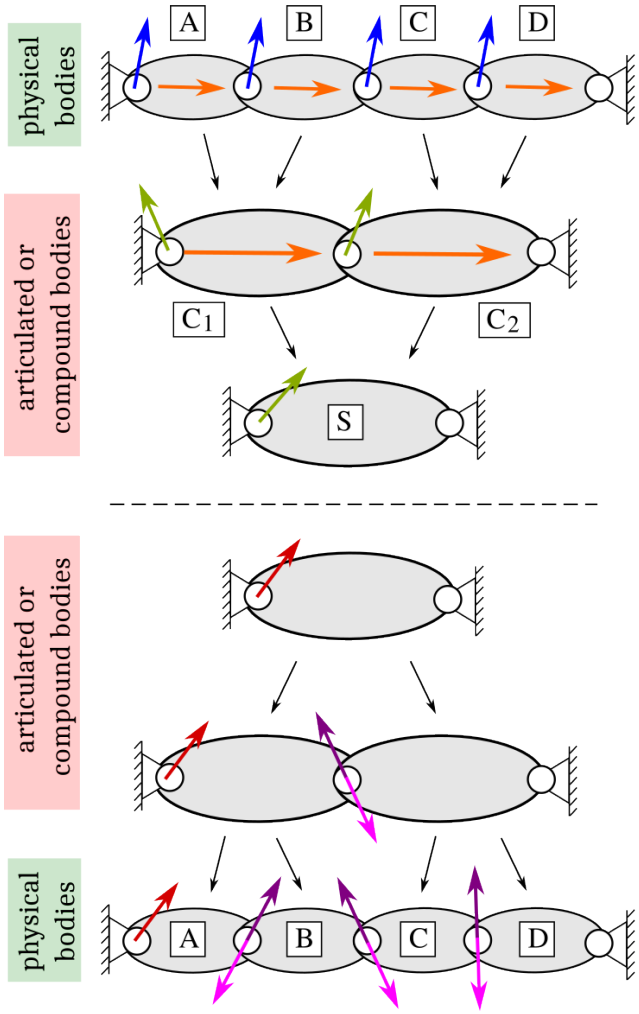
$$\overline{\mathbf{Q}}_1^B = \mathbf{Q}_1^B - \mathbf{S}_{12}^B \overline{\mathbf{Q}}_2^C = -\overline{\mathbf{Q}}_2^A, \tag{3.50}$$

where subassemblies A and B constitute parent-assembly C. Equation (3.50) describes the disassembly phase of the DCA algorithm. Let us remark that accumulated forces must be stored in the memory of a computer during both phases of the algorithm. The recursion terminates once the leaf-bodies are reached. Ultimately, this procedure achieves the goal described by eq. (3.46) for all bodies. Presented methodology is succinctly described in figure 3.6.

. . .

Now, let us utilize the derived approach to evaluate eq. (3.4b). Newton's second law states that a body's time rate of change of momentum equals the net force acting on its COM: $\dot{\mathbf{P}}_C^A = \mathbf{F}_C^A + \mathbf{R}_C^A$. Here, we divide the RHS into active forces $\mathbf{F}_C^A$ and reactions $\mathbf{R}_C^A$ originating from all constraints imposed on the body A. The derivative of momentum can be expressed at handle 1 by premultiplying Newton's law with $\mathbf{S}_{1C}^A$ and utilizing eq. (3.23). We substitute $\mathbf{P}_C^A = \mathbf{S}_{C1}^A \mathbf{P}_1^A$ and exploit the following relation $\mathbf{S}_{1C}\dot{\mathbf{S}}_{C1} = \dot{\mathbf{S}}_{C1}$, which can be confirmed by direct calculation. Finally, the dynamic equation becomes:

$$\dot{\mathbf{P}}_1^A = \mathbf{F}_1^A - \dot{\mathbf{S}}_{C1}^A \mathbf{P}_1^A + \mathbf{R}_1^A. \tag{3.51}$$



Accumulated force vectors are equavilent to generalized forces at the leaf-body level. One can evaluate spatial force vector for each body sparately.

Spatial force vectors are "accumulated" recursively. The shift matrix operator acts along the whole articulated body.

Articulated force of the root compound body is euqal to its accumulated force in handle 1 and equal to zero on the other end (handle 2)

Articulated forces are computed recursively from root node up to leaf bodies. Note that: $\overline{Q}_1^B = -\overline{Q}_2^A$

Articulated forces allow us to evaluate time derivatives of joint momenta in a parallel manner.

Figure 3.6: Stages of the DCA procedure for computing articulated forces

The derivative of eq. (3.22) can be written as: $\dot{\overline{\mathbf{P}}}_1^A = \dot{\overline{\mathbf{P}}}_1'^A + \mathbf{D}_i\dot{\boldsymbol{\sigma}}_i$, where the auxiliary variable $\dot{\overline{\mathbf{P}}}_1'^A = \mathbf{H}_i\dot{\widehat{\mathbf{p}}}_i + \dot{\mathbf{H}}_i\widehat{\mathbf{p}}_i + \dot{\mathbf{D}}_i\boldsymbol{\sigma}_i$. Hence, the derivative of eq. (3.29) reads as follows:

$$\begin{aligned}\dot{\mathbf{P}}_1^A &= \dot{\overline{\mathbf{P}}}_1^A + \mathbf{S}_{12}^A\dot{\overline{\mathbf{P}}}_2^A + \dot{\mathbf{S}}_{12}^A\overline{\mathbf{P}}_2^A,\\ &= \dot{\overline{\mathbf{P}}}_1'^A + \mathbf{D}_i\dot{\boldsymbol{\sigma}}_i + \mathbf{S}_{12}^A\left(\dot{\overline{\mathbf{P}}}_2'^A - \mathbf{D}_j\dot{\boldsymbol{\sigma}}_j\right) + \dot{\mathbf{S}}_{12}^A\overline{\mathbf{P}}_2^A.\end{aligned} \tag{3.52}$$

We substitute equation (3.52) into the left-hand side of eq. (3.51). Let us remark that $\dot{\boldsymbol{\sigma}}_i = \boldsymbol{\lambda}_i$, where $\boldsymbol{\lambda}$ signifies the values of reaction forces. Therefore, the terms involving quantities $\dot{\boldsymbol{\sigma}}$ cancel out with their counterparts $\mathbf{R}_1^A$:

$$\dot{\overline{\mathbf{P}}}_1'^A + \mathbf{S}_{12}^A\dot{\overline{\mathbf{P}}}_2'^A + \dot{\mathbf{S}}_{12}^A\overline{\mathbf{P}}_2^A = \mathbf{F}_1^A - \dot{\mathbf{S}}_{C1}^A\left(\overline{\mathbf{P}}_1^A + \mathbf{S}_{12}^A\overline{\mathbf{P}}_2^A\right). \tag{3.53}$$

Equation (3.53) can be reordered by taking into account additional relations between the derivatives of the shift matrix operator, i.e.: $\dot{\mathbf{S}}_{C1}\mathbf{S}_{12} = \dot{\mathbf{S}}_{C1}$ and $\dot{\mathbf{S}}_{C1} + \dot{\mathbf{S}}_{12} = \dot{\mathbf{S}}_{C2}$.

$$\dot{\overline{\mathbf{P}}}_1'^A + \mathbf{S}_{12}^A \dot{\overline{\mathbf{P}}}_2'^A = \mathbf{F}_1^A - \dot{\mathbf{S}}_{C1}^A \overline{\mathbf{P}}_1^A - \dot{\mathbf{S}}_{C2}^A \overline{\mathbf{P}}_2^A. \tag{3.54}$$

Note that all calculations from eq. (3.51) onward concern a single leaf-level body, hence generalized forces and accumulated forces are equivalent, and it is valid to equate these quantities: $\mathbf{F}_1^A = \mathbf{Q}_1^A$. The articulated force described by eq. (3.46) may be finally utilized transferring equation (3.54) into the following form:

$$\dot{\overline{\mathbf{P}}}_1'^A = \overline{\mathbf{Q}}_1^A - \dot{\mathbf{S}}_{C1}^A \overline{\mathbf{P}}_1^A - \dot{\mathbf{S}}_{C2}^A \overline{\mathbf{P}}_2^A + \mathbf{S}_{12}^A \big(\overline{\mathbf{Q}}_2^A - \dot{\overline{\mathbf{P}}}_2'^A\big). \tag{3.55}$$

It will be convenient to temporarily assume that links A and B constitute two adjacent, terminal bodies. Consequently, the last body in the kinematic chain can be described with the following relations: $\mathbf{Q}_1^B = \overline{\mathbf{Q}}_1^B$ and $\mathbf{P}_1^B = \overline{\mathbf{P}}_1^B$. This allows us to rewrite equation (3.51) for the body B as:

$$\dot{\mathbf{P}}_1^B = \dot{\overline{\mathbf{P}}}_1'^B + \cancel{\mathbf{D}_j \dot{\boldsymbol{\sigma}}_j} = \overline{\mathbf{Q}}_1^B - \dot{\mathbf{S}}_{C1}^B \overline{\mathbf{P}}_1^B + \cancel{\mathbf{R}_1^B}. \tag{3.56}$$

By applying the law of action-reaction and the relation $\dot{\mathbf{S}}_{1C}^B = -\dot{\mathbf{S}}_{C1}^B$ to eq. (3.56), we come up with the expression: $\overline{\mathbf{Q}}_2^A - \dot{\overline{\mathbf{P}}}_2'^A = \dot{\overline{\mathbf{P}}}_1'^B - \overline{\mathbf{Q}}_1^B \overset{(3.56)}{=} \dot{\mathbf{S}}_{1C}^B \overline{\mathbf{P}}_1^B = -\dot{\mathbf{S}}_{1C}^B \overline{\mathbf{P}}_2^A$, which upon substitution into eq. (3.55), converts it into:

$$\begin{aligned} \dot{\overline{\mathbf{P}}}_1'^A &= \overline{\mathbf{Q}}_1^A - \dot{\mathbf{S}}_{C1}^A \overline{\mathbf{P}}_1^A - \dot{\mathbf{S}}_{C2}^A \overline{\mathbf{P}}_2^A - \mathbf{S}_{12}^A \dot{\mathbf{S}}_{1C}^B \overline{\mathbf{P}}_2^A \\ &= \overline{\mathbf{Q}}_1^A - \dot{\mathbf{S}}_{C1}^A \overline{\mathbf{P}}_1^A - \big(\dot{\mathbf{S}}_{C2}^A + \dot{\mathbf{S}}_{1C}^B\big) \overline{\mathbf{P}}_2^A. \end{aligned} \tag{3.57}$$

Equation (3.57) presents a special case defined by the assumption specified at the beginning of this paragraph (under eq. (3.55)). It can be easily shown that the general case of the dynamic equation reads:

$$\dot{\overline{\mathbf{P}}}_1'^A = \overline{\mathbf{Q}}_1^A - \dot{\mathbf{S}}_{C1}^A \overline{\mathbf{P}}_1^A - \boldsymbol{\kappa}(A), \qquad \boldsymbol{\kappa}(A) = \sum_{k=A}^{des(A)} \big(\dot{\mathbf{S}}_{C2}^k + \dot{\mathbf{S}}_{1C}^{k+1}\big) \overline{\mathbf{P}}_2^k, \tag{3.58}$$

where the symbol $des(A)$ refers to the *descendants* of the body A (please refer to section 3.3.1 for more details on the nomenclature). The derivative of joint momentum is readily available, once we expand the symbol $\dot{\overline{\mathbf{P}}}_1'^A$ and project equation (3.58) onto the subspace $\mathbf{H}_i$:

$$\dot{\widehat{\mathbf{p}}}_i = \mathbf{H}_i^T \Big(\overline{\mathbf{Q}}_1^A - \dot{\mathbf{S}}_{C1}^A \overline{\mathbf{P}}_1^A - \boldsymbol{\kappa}(A) - \dot{\mathbf{H}}_i \widehat{\mathbf{p}}_i - \dot{\mathbf{D}}_i \boldsymbol{\sigma}_i \Big). \tag{3.59}$$

Since the velocity vector has been calculated in subsection 3.4.1, all quantities on the RHS of eq. (3.59) can be treated as known parameters. Let us also recall that under the

assumption of a tree-structured topology, the notation may follow a convention where the indices appearing in eq. (3.59) have the same value, i.e., $A = i$ (c.f. fig. 3.1), which simplifies the geometrical interpretation of the calculations.

The main advantage of the formula (3.59) over eq. (3.14b) is how clearly the concurrency is exposed. The quantity $\dot{\widehat{\mathbf{p}}}_i$ can be calculated separately for each body by a single thread over a multi-core machine. This may lead to significant computational gains when the number of bodies is sufficiently large. On the other hand, one may argue that the term $\boldsymbol{\kappa}$ is a potential obstacle when implementing eq. (3.59) in a parallel manner. A closer look reveals that this term represents a cumulative sum distributed over a kinematic chain, which also can be evaluated in a parallel fashion once velocities and impulses of reaction loads are obtained. A practical mechanism allowing for such an implementation is called a *reduction*.

Let us summarize the methodology presented in section 3.3. The procedure starts with known values of joint coordinates $\boldsymbol{\alpha}$ and joint momenta $\widehat{\mathbf{p}}$. One can extract the latter quantity from initial velocity by projecting equation (3.29) on the subspace $\mathbf{H}_i$, starting at the last body and recurring along the kinematic chain to the ground body. Subsection 3.4.1 shows the DCA procedure for the elimination of constraint impulsive loads by a recursive substitution of known coefficients of EOM (c.f. eq. (3.42)). Moreover, an opposite process is described that yields the unknown reactions (c.f. eq. (3.39)) and, ultimately, the velocities. A similar divide-and-conquer approach is exposed in subsection 3.4.2, where we show how one can distribute a physical force over the handles of a body. This procedure yields the articulated forces, which are explicitly invoked in eq. (3.59). The outcome of these calculations are the derivatives $\dot{\boldsymbol{\alpha}}$ and $\dot{\widehat{\mathbf{p}}}$, which can be passed to the ODE solver for a numerical integration.

3.5 Summary

This chapter has been focused on multiple approaches for solving the forward dynamics problem. First, in section 3.2, we introduced two rudimentary formulations based on the Hamilton's formalism. Presented approaches yield large, sparse matrices that can easily be computed systematically. We also pinpointed different possibilities of describing the velocity in the MBS. Concurrently, this section has outlayed a convention of notation that will be present throughout this work, which has been gathered in table 3.1

Furthermore, section 3.3 pertained to introducing basic concepts and terminology associated with a recursive approach to solving the forward problem of dynamics. Initially, we introduced certain subspaces and operators, such as allowable and constrained motion subspaces $\mathbf{H}_i, \mathbf{D}_i$ or shift matrix operator $\mathbf{S}_{12}^A$. These quantities have a straightforward

physical interpretation and allow one for many useful operations. For example, based on the motion subspaces and utilizing the projection methods, we were able to come up with joint-space equations of motion formulated globally.

Based on established knowledge, the remainder of this chapter has been devoted to the derivation of the recursive formula known as Hamiltonian Divide-and-Conquer Algorithm. This procedure builds a binary tree that reflects the topology of the MBS and traverses it from leaves (physical bodies) to a root node (abstract, compound *articulated body*) and back. Two distinct recursions have been described, which can be summarized by figures 3.5 (describing velocity-level recursion) and 3.6 (force-level recursion). Since the binary tree can be traversed concurrently, the HDCA procedure is highly amenable for parallelization. The delivered formulae allow one to implement a computer code that efficiently solves large scale multibody dynamics problems via parallel computing and is superior to classical formulations.

CHAPTER 4

ADJOINT SENSITIVITY ANALYSIS

4.1 Introduction

This chapter is devoted to the derivation and application of the adjoint method to Hamilton's equation introduced in the previous chapter. Section 4.2 expands the concepts presented in sec. 2.4.2; however, the performance measure now has a particular form:

$$J[\mathbf{q}(\cdot), \mathbf{v}(\cdot), \mathbf{b}] = \int_0^{t_f} h(t, \mathbf{q}, \mathbf{v}, \mathbf{b})\, \mathrm{d}t + S(\mathbf{q}(t_f), \mathbf{v}(t_f)). \tag{4.1}$$

The author has presented a detailed derivation of the adjoint system based on the Hamiltonian formalism in ref. [79], where the integrand h takes into account other, more involved dependencies, such as reaction loads or accelerations. However, such a generalization would significantly obscure the main picture of this thesis. The concepts presented herein remain the same as in the paper; however, the underlying equations of motion are stated in the form of eqs. (3.14) instead of eqs. (3.9). This distinction is motivated by the fact that the mass matrix of a single body, $\mathbf{M}^{\#A}$, is non-invertible in the case of the latter formulation. The property of a local mass matrix invertibility is crucial when one aims at establishing divide-and-conquer relations. Consequently, the usage of the formulation (3.14) raises a new problem: the non-integrability of the angular velocity and its variation. A remedy for this issue is proposed in section 4.2 and appendix A.

Section 4.3 presents a novel way of reducing the adjoint system from a set of DAEs into the ODEs [81]. We begin by introducing independent adjoint variables defined in the multibody system's joint-space. The underlying idea for the subsequent derivations is to establish spatial relations between absolute adjoint variables defined in sec. 4.2. Certain elegant analogies arise from these calculations. Ultimately, the adjoint system formulated

as a set of ODEs is obtained by combining the resultant relationships derived in sec. 4.3 with the projection methods.

Finally, the relations derived in sec. 4.3 are put to use in section 4.5, where we introduce a DCA scheme for the adjoint system based on Hamilton's equations of motion. We utilize the knowledge introduced in section 3.4.1 and lay out analogies that appear between forward and adjoint problems when one investigates the divide-and-conquer paradigm.

4.2 Hamiltonian-based adjoint method (global formulation)

This section investigates applying the adjoint method to multibody dynamics expressed in terms of Hamilton's canonical equations. Let us rewrite the underlying EOM (3.14), derived in section 3.2, to explicitly introduce control variables $\mathbf{b}$, such as design parameters and discretized input functions:

$$\begin{bmatrix} \mathbf{M}(\mathbf{q},\mathbf{b}) & \mathbf{C}^T(\mathbf{q},\mathbf{b}) \\ \mathbf{C}(\mathbf{q},\mathbf{b}) & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \boldsymbol{\sigma} \end{bmatrix} = \begin{bmatrix} \mathbf{p}^* \\ \mathbf{0} \end{bmatrix}, \tag{4.2a}$$

$$\dot{\mathbf{p}}^* = \mathbf{f}(\mathbf{q},\mathbf{v},\mathbf{b},t) + \dot{\mathbf{C}}^T(\mathbf{q},\mathbf{v},\mathbf{b})\boldsymbol{\sigma}. \tag{4.2b}$$

Equations of motion must be fulfilled throughout the optimization process, hence it is valid to treat them as constraints imposed on the design variables $\mathbf{b}$. The key principle of the adjoint method is the inclusion of these constraints to the generic cost functional (4.1):

$$\overline{J} = \int_0^{t_f} \left[h + \boldsymbol{\eta}^T(\mathbf{p}^* - \mathbf{M}\mathbf{v} - \mathbf{C}^T\boldsymbol{\sigma}) + \boldsymbol{\xi}^T(\dot{\mathbf{p}}^* - \mathbf{f} - \dot{\mathbf{C}}^T\boldsymbol{\sigma}) - \boldsymbol{\mu}^T\dot{\boldsymbol{\Phi}} \right] \mathrm{d}t + S. \tag{4.3}$$

The quantities $\boldsymbol{\eta} = \boldsymbol{\eta}(t) \in \mathcal{R}^{n_\beta}$, $\boldsymbol{\xi} = \boldsymbol{\xi}(t) \in \mathcal{R}^{n_\beta}$, and $\boldsymbol{\mu} = \boldsymbol{\mu}(t) \in \mathcal{R}^{m}$ are arbitrary Lagrange multipliers, also known as *adjoint (costate) variables*, that enforce the system equations (4.2). We can note that the gradient of the functional (4.1) is equal to the gradient of the augmented functional (4.3), i.e.: $\nabla_{\mathbf{b}} J = \nabla_{\mathbf{b}} \overline{J}$ along any solution of the system state equations. According to the fundamental theorem of the calculus of variations, the necessary condition for an extremum of the expression (4.3) is fulfilled when the first variation $\delta\overline{J}$ vanishes. The variation of the extended functional can be written in the following form:

$$
\begin{aligned}
\delta\overline{J} = \int_0^{t_f} \Big[& \Big(h_{\mathbf{b}} - \boldsymbol{\eta}^T (\mathbf{M}\mathbf{v})_{\mathbf{b}} - \boldsymbol{\eta}^T (\mathbf{C}^T \boldsymbol{\sigma})_{\mathbf{b}} - \boldsymbol{\xi}^T \mathbf{f}_{\mathbf{b}} - \boldsymbol{\xi}^T (\dot{\mathbf{C}}^T \boldsymbol{\sigma})_{\mathbf{b}} - \boldsymbol{\mu}^T \dot{\boldsymbol{\Phi}}_{\mathbf{b}} \Big) \delta\mathbf{b} + \\
& \boldsymbol{\eta}^T \Big(\delta\mathbf{p}^* - \mathbf{M}\, \delta\mathbf{v} - (\mathbf{M}\mathbf{v})_{\mathbf{s}}\, \delta\mathbf{s} - \mathbf{C}^T\, \delta\boldsymbol{\sigma} - (\mathbf{C}^T \boldsymbol{\sigma})_{\mathbf{s}}\, \delta\mathbf{s} \Big) + \\
& \boldsymbol{\xi}^T \Big(\delta\dot{\mathbf{p}}^* - \mathbf{f}_{\mathbf{v}}\, \delta\mathbf{v} - \mathbf{f}_{\mathbf{s}}\, \delta\mathbf{s} - \dot{\mathbf{C}}^T\, \delta\boldsymbol{\sigma} - (\dot{\mathbf{C}}^T \boldsymbol{\sigma})_{\mathbf{v}}\, \delta\mathbf{v} - (\dot{\mathbf{C}}^T \boldsymbol{\sigma})_{\mathbf{s}}\, \delta\mathbf{s} \Big) - \\
& \boldsymbol{\mu}^T \Big(\dot{\boldsymbol{\Phi}}_{\mathbf{v}}\, \delta\mathbf{v} - \dot{\boldsymbol{\Phi}}_{\mathbf{s}}\, \delta\mathbf{s} \Big) + h_{\mathbf{s}}\, \delta\mathbf{s} + h_{\mathbf{v}}\, \delta\mathbf{v} \Big] \, \mathrm{d}t + S_{\mathbf{v}}\, \delta\mathbf{v} + S_{\mathbf{s}}\, \delta\mathbf{s},
\end{aligned}
\tag{4.4}
$$

where the subindices denote a partial differential operator. The subindex $\mathbf{s}$ represents a partial derivative involving translational and angular components. The first one is straightforward, denoting a partial derivative with respect to the Cartesian coordinates $\mathbf{r}$. The angular portion of the expression can be treated as coefficients standing next to the virtual rotation $\delta\boldsymbol{\pi}'$ after computing the quantity variation before $\mathbf{s}$. Since the latter quantity is non-integrable, the Jacobian matrices calculated with respect to $\delta\boldsymbol{\pi}'$ (or $\mathbf{s}$) should be interpreted only as coefficients of a certain variational relation, rather than partial derivatives. Subsequently, the operator $\delta\mathbf{s}$ refers to the virtual displacement and virtual rotation of the premultiplied expression. Variational quantities introduced in eq. (4.4) can be written explicitly as (quantity $\mathbf{q}$ is added for completeness):

$$
\mathbf{q} = \begin{bmatrix} \vdots \\ \mathbf{r}_i \\ \widehat{\boldsymbol{\eta}}_i \\ \vdots \end{bmatrix} \Rightarrow \delta\mathbf{q} = \begin{bmatrix} \vdots \\ \delta\mathbf{r}_i \\ \delta\widehat{\boldsymbol{\eta}}_i \\ \vdots \end{bmatrix}, \quad \delta\mathbf{s} = \begin{bmatrix} \vdots \\ \delta\mathbf{r}_i \\ \delta\boldsymbol{\pi}'_i \\ \vdots \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} \vdots \\ \dot{\mathbf{r}}_i \\ \boldsymbol{\omega}'_i \\ \vdots \end{bmatrix} \Rightarrow \delta\mathbf{v} = \begin{bmatrix} \vdots \\ \delta\dot{\mathbf{r}}_i \\ \delta\boldsymbol{\omega}'_i \\ \vdots \end{bmatrix}. \tag{4.5}
$$

Similarly to eq. (3.11), the following relationships are also true: $\delta\mathbf{q} = \mathbf{L}_h^T\, \delta\mathbf{s}$, $\delta\mathbf{s} = \mathbf{L}_d\, \delta\mathbf{q}$.

Let us note that angular velocity $\boldsymbol{\omega}'$ and its variation are non-integrable. In the following step we aim to integrate equation (4.4) by parts, which will require to substitute the variation of the angular velocity with an integrable quantity. To this end, we argue that equation (4.4) can be simplified by applying the following relation: $\delta\mathbf{v} = \delta\dot{\mathbf{s}}$ (or $\delta\boldsymbol{\omega}' = \delta\dot{\boldsymbol{\pi}}'$ if we focused on rotational part only). We derive equation (A.7) (or, more specifically, eq. (A.5)) in appendix A, which is a more general form of this substitution. Furthermore, we motivate why the second term on the RHS of eq. (A.7) must be dropped.

A general formula for integration by parts has been shown in eq. (2.9). As an example, the simplest component in the performance measure (4.4) can be expanded in the following way:

$$
\int_0^{t_f} \boldsymbol{\xi}^T \delta\dot{\mathbf{p}}^*\, \mathrm{d}t = -\int_0^{t_f} \dot{\boldsymbol{\xi}}^T \delta\mathbf{p}^*\, \mathrm{d}t + \boldsymbol{\xi}^T \delta\mathbf{p}^*\big|_{t_f} - \cancel{\boldsymbol{\xi}^T \delta\mathbf{p}^*\big|_{t_0}}. \tag{4.6}
$$

Here, we assume that initial conditions for a multibody system are prescribed, fi.e.: $\mathbf{q}(0) = \mathbf{q}_0$, $\mathbf{p}^*(0) = \mathbf{p}_0^*$, which yields: $\delta\mathbf{p}^*|_{t_0} = \mathbf{0}$. Conversely, the initial state of the MBS may be a part of the optimization process. In such a case, relation (4.6) must be extended to take this fact into account. This approach is studied in greater detail in ref. [77].

The final form of $\delta\overline{J}$ can be calculated by integrating quantities $\delta\,\dot{\mathbf{p}}^*$ and $\delta\mathbf{v} := \delta\dot{\mathbf{s}}$ by parts accordingly to the scheme presented in eq (4.6). Ultimately, equation (4.4) can be rewritten as:

$$\begin{aligned}\delta\overline{J} = \int_0^{t_f} \Big[&\left(\dot{\boldsymbol{\eta}}^T\mathbf{M} + \dot{\boldsymbol{\xi}}^T\mathbf{A}^T + \dot{\boldsymbol{\mu}}^T\mathbf{C} + \mathbf{r}_{\mathcal{A}}^T\right)\,\delta\mathbf{s} - \left(\boldsymbol{\eta}^T\mathbf{C}^T + \boldsymbol{\xi}^T\dot{\mathbf{C}}^T\right)\delta\boldsymbol{\sigma} + \\ &\left(\boldsymbol{\eta}^T - \dot{\boldsymbol{\xi}}^T\right)\delta\mathbf{p}^* + \left(h_{\mathbf{b}} - \boldsymbol{\eta}^T\,\breve{\mathbf{p}}_{\mathbf{b}}^* - \boldsymbol{\xi}^T\,\breve{\dot{\mathbf{p}}}_{\mathbf{b}}^* - \boldsymbol{\mu}^T\dot{\boldsymbol{\Phi}}_{\mathbf{b}}\right)\delta\mathbf{b}\Big]\,\mathrm{d}t + \\ &\left(S_{\mathbf{s}} + h_{\mathbf{v}} - \boldsymbol{\eta}^T\mathbf{M} - \boldsymbol{\xi}^T\mathbf{A} - \boldsymbol{\mu}^T\mathbf{C}\right)\,\delta\mathbf{s}\big|_{t_f} + \left(\boldsymbol{\xi}^T\right)\delta\mathbf{p}^*\big|_{t_f} + S_{\mathbf{v}}\delta\mathbf{v}|_{t_f}\,,\end{aligned} \tag{4.7}$$

where the compact structure is achieved by defining some additional quantities:

$$\begin{aligned}&\breve{\mathbf{p}}_{\mathbf{b}}^* = \left(\mathbf{M}\mathbf{v} + \mathbf{C}^T\boldsymbol{\sigma}\right)_{\mathbf{b}}, \quad \breve{\dot{\mathbf{p}}}_{\mathbf{b}}^* = \left(\mathbf{f} + \dot{\mathbf{C}}^T\boldsymbol{\sigma}\right)_{\mathbf{b}},\\ &\breve{\mathbf{p}}_{\mathbf{s}}^* = \left(\mathbf{M}\mathbf{v} + \mathbf{C}^T\boldsymbol{\sigma}\right)_{\mathbf{s}}, \quad \breve{\dot{\mathbf{p}}}_{\mathbf{s}}^* = \left(\mathbf{f} + \dot{\mathbf{C}}^T\boldsymbol{\sigma}\right)_{\mathbf{s}},\\ &\breve{\mathbf{p}}_{\mathbf{v}}^* = \mathbf{M}, \qquad\qquad \mathbf{A}^T = \breve{\dot{\mathbf{p}}}_{\mathbf{v}}^* = \left(\mathbf{f} + \dot{\mathbf{C}}^T\boldsymbol{\sigma}\right)_{\mathbf{v}},\\ &\mathbf{r}_{\mathcal{A}}^T = h_{\mathbf{s}} - \frac{\mathrm{d}}{\mathrm{d}t}h_{\mathbf{v}} - \boldsymbol{\eta}^T\left(\breve{\mathbf{p}}_{\mathbf{s}}^* - \dot{\mathbf{M}}\right) - \boldsymbol{\xi}^T\left(\breve{\dot{\mathbf{p}}}_{\mathbf{s}}^* - \frac{\mathrm{d}}{\mathrm{d}t}\breve{\dot{\mathbf{p}}}_{\mathbf{v}}^*\right).\end{aligned} \tag{4.8}$$

One can associate the breve accent $\breve{}$ in the eqs. (4.8) as an indication of the explicit partial derivative of the equations of motion (4.2) with respect to the appropriate variable, implied by the lower index. We investigate the origin of relations (4.8) in appendix B in further detail.

At this point, the adjoint variables are arbitrary functions of time. In the following step, we express the variation of the cost functional (4.3) solely in terms of the variation of the design variables vector $\mathbf{b}$. This approach releases us from the need of calculating all implicit dependencies between state and design variables. To this end, we equate the terms multiplied by the dependent variations $\delta\mathbf{s}$, $\delta\mathbf{p}^*$, and $\delta\boldsymbol{\sigma}$ to zero. Consequently, the variation of extended performance measure is reduced to:

$$\delta\overline{J} = \int_0^{t_f}\left(h_{\mathbf{b}} - \boldsymbol{\eta}^T\,\breve{\mathbf{p}}_{\mathbf{b}}^* - \boldsymbol{\xi}^T\,\breve{\dot{\mathbf{p}}}_{\mathbf{b}}^* - \boldsymbol{\mu}^T\dot{\boldsymbol{\Phi}}_{\mathbf{b}}\right)\delta\mathbf{b}\,\mathrm{d}t, \tag{4.9}$$

The variation of performance measure is now solely dependent on the variation of the design variables; however, equation (4.9) is valid only for a uniquely defined set of adjoint variables. The necessary conditions for the appropriate values follow directly from eq. (4.7)

and have the following form. First, let us equate to zero the terms under the integral next to the variations $\delta\mathbf{s}, \delta\boldsymbol{\sigma}, \delta\mathbf{p}^*$, respectively:

$$\mathbf{M}\dot{\boldsymbol{\eta}} + \mathbf{C}^T\dot{\boldsymbol{\mu}} + \mathbf{A}\dot{\boldsymbol{\xi}} + \mathbf{r}_{\mathcal{A}} = \mathbf{0}, \tag{4.10a}$$

$$\mathbf{C}\boldsymbol{\eta} + \dot{\mathbf{C}}\boldsymbol{\xi} = \mathbf{0}, \tag{4.10b}$$

$$\boldsymbol{\eta} - \dot{\boldsymbol{\xi}} = \mathbf{0}. \tag{4.10c}$$

Equations (4.10) constitute a set of first order differential-algebraic equations with differential index 2. One of the objectives of this thesis is to propose an efficient algorithm for solving the adjoint system described by eq. (4.10). To this end, a significant computational benefit can be gained by writing eqs. (4.10) as a set of ordinary–differential equations and reducing their differentiation index. The substitution of $\dot{\boldsymbol{\xi}}$ from eq. (4.10c) into a time derivative of eq. (4.10b) and into eq. (4.10a) yields the following system of equations:

$$\dot{\boldsymbol{\xi}} = \boldsymbol{\eta}. \tag{4.11a}$$

$$\begin{bmatrix} \mathbf{M} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\eta}} \\ \dot{\boldsymbol{\mu}} \end{bmatrix} = \begin{bmatrix} -\mathbf{r}_{\mathcal{A}} - \mathbf{A}\boldsymbol{\eta} \\ -2\dot{\mathbf{C}}\boldsymbol{\eta} - \ddot{\mathbf{C}}\boldsymbol{\xi} \end{bmatrix} = \begin{bmatrix} -\mathbf{r} \\ -\mathbf{r}_c \end{bmatrix}. \tag{4.11b}$$

A direct comparison with equation (4.2) shows that the matrix of coefficients is the same as in eq. (4.11b). This fact will be exploited to increase the efficiency of the formulation. It should be noted that eqs. (4.11) have a relatively simple structure, and the only arising challenge lies in an accurate calculation of its right-hand side, precisely the term $\mathbf{r}$. Appendix B is devoted to presenting these calculations in greater detail.

Let us now consider the terminal expressions of eq. (4.7). They establish boundary conditions for the adjoint variables. The current form of eq. (4.7) allows only to prescribe the terminal configuration of the system. This is dictated by the fact that $S_{\mathbf{v}}$ must be equal to zero to obtain the form described by eq. (4.9), which implies no dependency between terminal cost and the velocity. Of course, this is against the assumption specified earlier in eq. (4.1). This issue can be resolved by defining additional adjoint variables $\boldsymbol{\nu} \in \mathcal{R}^{n_v}$, $\boldsymbol{\varepsilon} \in \mathcal{R}^m$ at terminal time and calculating the variation of equation (4.2a) for t_f:

$$\begin{aligned} \delta\Big[\boldsymbol{\nu}^T\left(\mathbf{p}^* - \mathbf{M}\mathbf{v} - \mathbf{C}^T\boldsymbol{\sigma}\right)\big|_{t_f} - \boldsymbol{\varepsilon}^T\left(\mathbf{C}\mathbf{v}\right)\big|_{t_f}\Big] = &-\left(\boldsymbol{\nu}^T\mathbf{M} + \boldsymbol{\varepsilon}^T\mathbf{C}\right)\delta\mathbf{v}\big|_{t_f} + \\ &\boldsymbol{\nu}^T\delta\mathbf{p}^*\big|_{t_f} - \boldsymbol{\nu}^T\mathbf{C}^T\delta\boldsymbol{\sigma}\big|_{t_f} - \boldsymbol{\nu}^T\,\widetilde{\mathbf{p}}_{\mathbf{s}}^*\;\delta\mathbf{s}\Big|_{t_f} - \boldsymbol{\varepsilon}^T(\mathbf{C}\mathbf{v})_{\mathbf{s}}\;\delta\mathbf{s}\big|_{t_f}. \end{aligned} \tag{4.12}$$

Since the quantities in the parentheses on the left-hand side are equal to zero, it is valid to add eq. (4.12) to the performance measure (4.3) and recalculate its total variation.

Effectively, this appends the variation (4.7) with the RHS of eq. (4.12), yielding the following boundary conditions for the adjoint system:

$$\begin{bmatrix} \mathbf{M} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\nu} \\ \boldsymbol{\varepsilon} \end{bmatrix}_{t_f} = \begin{bmatrix} S_{\mathbf{v}}^T \\ \mathbf{0} \end{bmatrix}_{t_f}, \tag{4.13a}$$

$$\boldsymbol{\xi}|_{t_f} = -\boldsymbol{\nu}, \tag{4.13b}$$

$$\begin{bmatrix} \mathbf{M} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta} \\ \boldsymbol{\mu} \end{bmatrix}_{t_f} = \begin{bmatrix} (S_{\mathbf{s}} + h_{\mathbf{v}})^T + \left((\widetilde{\mathbf{p}}_{\mathbf{s}}^*)^T - \mathbf{A} \right) \boldsymbol{\xi} - (\mathbf{C}\mathbf{v})_{\mathbf{s}}^T \boldsymbol{\varepsilon} \\ -\dot{\mathbf{C}}\boldsymbol{\xi} \end{bmatrix}_{t_f}. \tag{4.13c}$$

Once the boundary values for the multipliers are evaluated via eqs. (4.13), the set of differential-algebraic equations (4.11b) may be solved backwards in time to obtain $\boldsymbol{\eta} = \boldsymbol{\eta}(t)$, $\boldsymbol{\xi} = \boldsymbol{\xi}(t)$, and $\boldsymbol{\mu} = \boldsymbol{\mu}(t)$ at every time instant. When solving the adjoint system backwards in time, it will be convenient to introduce a new independent variable, i.e.: $\tau = t_f - t, \tau \in [0, t_f]$. Therefore, we have to take into account the following relation: $\frac{d}{dt} = \frac{d}{d\tau}\frac{d\tau}{dt} = -\frac{d}{d\tau}$:

$$\dot{\boldsymbol{\xi}} = -\boldsymbol{\eta} \tag{4.14a}$$

$$\begin{bmatrix} \mathbf{M} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\eta}} \\ \dot{\boldsymbol{\mu}} \end{bmatrix} = \begin{bmatrix} \mathbf{r} \\ \mathbf{r}_c \end{bmatrix}. \tag{4.14b}$$

Equations (4.14) are solved for the unknown derivatives of the adjoint variables from $\tau = 0$ to $\tau = t_f$. Since independent variable τ marches in the reverse direction to the physical time, state variables, which are known coefficients of this equation, must be evaluated according to the following formula: $\mathbf{y}(\tau) = \mathbf{y}(t_f - t)$.

It is important to note, that state parameters must be stored in memory while solving the adjoint system (4.14b). Moreover, certain large-scale applications (e.g. the ones that involve finite element discretizations) would require to store significant amounts of simulation data to be used in the backward integration of the adjoint system. However, this procedure might be impractical or even infeasible. In such cases, a checkpointing scheme usually offers a compromise between execution time and total storage requirements [109]. The idea is to record state variables only at certain time instances (checkpoints) during the forward integration phase. This generates a coarse mesh of checkpoints which allows one for a quick reintegration of the EOM whenever the state variables are required during backward integration of the adjoint system. The cost of employing the checkpointing approach is one additional forward integration of EOM at the most, where the integration must be carried out in chunks.

Upon the solution of the system of adjoint equation, the variation of the performance measure reduces to the form presented in eq. (4.9), which relates the independent variation $\delta\mathbf{b}$ to the variation $\delta\overline{J}$. Therefore, the explicit formula for the gradient of the cost functional with respect to design variables can be written as:

$$(\nabla_{\mathbf{b}} J)^T = \int_0^{t_f} \left(h_{\mathbf{b}} - \boldsymbol{\eta}^T \widetilde{\mathbf{p}}_{\mathbf{b}}^* - \boldsymbol{\xi}^T \widetilde{\dot{\mathbf{p}}}_{\mathbf{b}}^* - \boldsymbol{\mu}^T \dot{\boldsymbol{\Phi}}_{\mathbf{b}} \right) \mathrm{d}t. \tag{4.15}$$

It should be noted that the expression (4.15) might be significantly simplified if we assume that only generalized forces are dependent on the design variables, which may represent spring or damping coefficients. In such case, eq. (4.15) reduces to a much simpler form, which can be evaluated numerically as:

$$(\nabla_{\mathbf{b}} J)^T = \int_0^{t_f} \left(h_{\mathbf{b}} - \boldsymbol{\xi}^T \mathbf{f}_{\mathbf{b}} \right) dt. \tag{4.16}$$

Equations (4.15), (4.16) refer to situations when $\mathbf{b}$ describes a vector of static design parameters. Conversely, design parameters may constitute sampled values that correspond to time-discretization. Hence, the resultant number of design parameters will usually be much larger than in the case of time independent design variables. The formula for a gradient, however, will remain similar to that demonstrated in eq. (4.16), i.e.: $(\nabla_{\mathbf{b}} J)^T = \left(h_{\mathbf{b}} - \boldsymbol{\xi}^T \mathbf{f}_{\mathbf{b}} \right) \Delta t$, where Δt is the time step.

4.3 Independent adjoint variables

In this section, we would like to exploit the advantageous properties of Hamiltonian formulation to introduce the joint-space adjoint variables. This derivation is based on two concepts: spatial dependency of absolute adjoint variables across a single body and the relation between absolute coordinate and joint coordinate formulations of EOM. By combining both relationships, we will be able to properly define a set of joint-space, independent adjoint variables.

Let us investigate the former concept first. Equations of motion can be formed and solved in various coordinate systems, which can be fixed at different geometric points of bodies within the MBS. Specific points, such as COM or body handles (c.f. sec. 3.3.1), are more convenient to consider than others since certain expressions significantly simplify. Accordingly, eq. (3.14), which describes the dynamics expressed in absolute coordinates, can be rewritten and represented at handle 1 for each body in the system:

$$\begin{aligned}\boldsymbol{\phi}_1 &\equiv \mathbf{p}_1^* - \mathbf{M}_1\mathbf{v}_1 - \mathbf{C}_1^T\boldsymbol{\sigma} = \mathbf{0},\\ \dot{\boldsymbol{\phi}}_1 &\equiv \dot{\mathbf{p}}_1^* - \mathbf{f}_1 - \dot{\mathbf{C}}_1^T\boldsymbol{\sigma} = \mathbf{0}.\end{aligned} \tag{4.17}$$

Equation (3.14), which is a special case of eq. (4.17), has been formulated in the COM of the bodies, hence it does not include any subindex. The calculus introduced in section 3.3.1 can be generalized to the stacked vector format by introducing the global shift matrix:

$$\mathbf{S}_{12} = \begin{bmatrix} \mathbf{S}_{12}^A & & & \\ & \mathbf{S}_{12}^B & & \\ & & \ddots & \\ & & & \mathbf{S}_{12}^{N_b} \end{bmatrix}, \qquad \mathbf{S}_{12}^A = \begin{bmatrix} \mathbf{1}_{3\times3} & \mathbf{0}_{3\times3} \\ \widetilde{\mathbf{s}}_{12}^{(A)}\mathbf{A}_A^T & \mathbf{1}_{3\times3} \end{bmatrix}. \tag{4.18}$$

Since the quantities described by equation (4.17) are linear and angular momenta with their derivatives (equal to sum of forces and torques), we can distinguish the following compact relations:

$$\boldsymbol{\phi}_1 = \mathbf{S}_{12}\boldsymbol{\phi}_2, \quad \text{and} \quad \dot{\boldsymbol{\phi}}_1 = \mathbf{S}_{12}\dot{\boldsymbol{\phi}}_2 + \dot{\mathbf{S}}_{12}\boldsymbol{\phi}_2. \tag{4.19}$$

Let us now rewrite eq. (4.3) with the dynamic equations expressed explicitly in handle 2:

$$\overline{\overline{J}} = \int_0^{t_f} \left[h + \boldsymbol{\eta}_2^T\boldsymbol{\phi}_2 + \boldsymbol{\xi}_2^T\dot{\boldsymbol{\phi}}_2 - \boldsymbol{\mu}^T\dot{\boldsymbol{\Phi}} \right] \mathrm{d}t + S. \tag{4.20}$$

Concurrently, one can rewrite eq. (4.3) once again and express the EOM in handle 1. By employing relations registered in eq. (4.19), we can additionally expand the right-hand side of the performance measure in the following way:

$$\overline{\overline{J}} = \int_0^{t_f} \left[h + \boldsymbol{\eta}_1^T\boldsymbol{\phi}_1 + \boldsymbol{\xi}_1^T\dot{\boldsymbol{\phi}}_1 - \boldsymbol{\mu}^T\dot{\boldsymbol{\Phi}} \right] \mathrm{d}t + S \tag{4.21a}$$

$$= \int_0^{t_f} \left[h + \underbrace{\left(\boldsymbol{\eta}_1^T\mathbf{S}_{12} + \boldsymbol{\xi}_1^T\dot{\mathbf{S}}_{12}\right)}_{\boldsymbol{\eta}_2^T}\boldsymbol{\phi}_2 + \underbrace{\boldsymbol{\xi}_1^T\mathbf{S}_{12}}_{\boldsymbol{\xi}_2^T}\dot{\boldsymbol{\phi}}_2 - \boldsymbol{\mu}^T\dot{\boldsymbol{\Phi}} \right] \mathrm{d}t + S, \tag{4.21b}$$

Equations (4.20) and (4.21b) have the same meaning since they refer to the identical phenomenon, where the enforced dynamical system is expressed in handle 2 for each body. On the other hand, the structure of these equations is different, exposing functional relationships between adjoint variables that can be interpreted as spatial relations along a rigid body:

$$\boldsymbol{\xi}_2 = \mathbf{S}_{12}^T \boldsymbol{\xi}_1 \tag{4.22a}$$

$$\boldsymbol{\eta}_2 = \mathbf{S}_{12}^T \boldsymbol{\eta}_1 + \dot{\mathbf{S}}_{12}^T \boldsymbol{\xi}_1 \tag{4.22b}$$

Equations (4.22) establish relations between adjoint variables $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ corresponding to different handles. One can see that the variable $\boldsymbol{\xi}$ transforms precisely in the same way as spatial velocity (c.f. eq. (3.18)). Consequently, $\boldsymbol{\eta}$ behaves similarly to the spatial acceleration, since $\boldsymbol{\eta} = \dot{\boldsymbol{\xi}}$. These analogies will be extensively used in further derivations.

Now, let us look into the second concept of the derivation, i.e. the relation between absolute coordinate and joint coordinate formulations of the EOM. Specifically, joint-space equations of motion (3.27) were obtained by projecting eqs. (3.9) onto the global motion subspace $\mathbf{H}$ introduced in sec. 3.3.2. Let us rewrite these equations assuming that the MBS has an open-loop kinematic chain. Moreover, here we drop the subindex specifying the origin of the coordinate frames in which EOM are solved:

$$\begin{aligned} \boldsymbol{\varphi} &\equiv \widehat{\mathbf{p}} - \widehat{\mathbf{M}}\boldsymbol{\beta} = \mathbf{0}, \\ \dot{\boldsymbol{\varphi}} &\equiv \dot{\widehat{\mathbf{p}}} - \mathbf{H}^T\mathbf{f} - \dot{\mathbf{H}}^T\mathbf{M}\mathbf{H}\boldsymbol{\beta} = \mathbf{0}. \end{aligned} \tag{4.23}$$

The relation between equations (4.17) and eqs. (4.23) is as follows:

$$\boldsymbol{\varphi} = \mathbf{H}^T\boldsymbol{\phi} \quad \Rightarrow \quad \dot{\boldsymbol{\varphi}} = \mathbf{H}^T\dot{\boldsymbol{\phi}} + \dot{\mathbf{H}}^T\boldsymbol{\phi}. \tag{4.24}$$

Similarly to the derivations shown in the previous paragraph, we can deduce valuable relationships from specific manipulations on the performance measure. Instead of augmenting the performance measure (4.1) with the absolute coordinate equations (4.17), we utilize the joint-space equations (4.23) and invoke eq. (4.24) to come up with the following relationships:

$$\begin{aligned} \overline{J} &= \int_0^{t_f} \left[h + \widehat{\boldsymbol{\eta}}^T\boldsymbol{\varphi} + \widehat{\boldsymbol{\xi}}^T\dot{\boldsymbol{\varphi}} - \boldsymbol{\mu}^T\dot{\boldsymbol{\Phi}} \right] \mathrm{d}t + S \\ &= \int_0^{t_f} \Big[h + \underbrace{\left(\widehat{\boldsymbol{\eta}}^T\mathbf{H}^T + \widehat{\boldsymbol{\xi}}^T\dot{\mathbf{H}}^T\right)}_{\boldsymbol{\eta}^T}\boldsymbol{\phi} + \underbrace{\left(\widehat{\boldsymbol{\xi}}^T\mathbf{H}^T\right)}_{\boldsymbol{\xi}^T}\dot{\boldsymbol{\phi}} - \boldsymbol{\mu}^T\dot{\boldsymbol{\Phi}} \Big] \mathrm{d}t + S. \end{aligned} \tag{4.25}$$

The variables $\widehat{\boldsymbol{\eta}} \in \mathcal{R}^{n_\beta}$ and $\widehat{\boldsymbol{\xi}} \in \mathcal{R}^{n_\beta}$ denote independent adjoint variables defined in the joint-space. By comparing equation (4.25) with eq. (4.3), they are related to their absolute-coordinate counterparts with the following relation:

$$\boldsymbol{\xi} = \mathbf{H}\widehat{\boldsymbol{\xi}}, \tag{4.26a}$$

$$\boldsymbol{\eta} = \mathbf{H}\widehat{\boldsymbol{\eta}} + \dot{\mathbf{H}}\widehat{\boldsymbol{\xi}}. \tag{4.26b}$$

The equations shown in (4.26) suggest that $\widehat{\boldsymbol{\eta}} = \dot{\widehat{\boldsymbol{\xi}}}$, which is a joint-space equivalent of equation (4.11a). This relation is indeed true and will be confirmed more rigorously in section 4.4. Let us insert eq. (4.10c) into eq. (4.10b) and integrate the outcome by parts. The result reads:

$$\boldsymbol{\Psi} \equiv \mathbf{C}\boldsymbol{\xi} = \mathbf{0}, \tag{4.27}$$

which is the implicit "velocity-level" constraint equation imposed on the adjoint variable $\boldsymbol{\xi}$. Similarly, equation (4.26a) represents the same kinematic constraint formulated explicitly. Ultimately, eqs. (4.22a), (4.27) (4.26a) reveal an interesting property of the adjoint variable $\boldsymbol{\xi}$. Apparently, this quantity is, to some extent, analogous to the spatial velocity vector $\mathbf{v}$. Table 4.1 summarizes the observed analogies.

Table 4.1: Analogies between spatial velocity vector $\mathbf{v} \in \mathcal{R}^{n_\beta}$ and the adjoint variable $\boldsymbol{\xi} \in \mathcal{R}^{n_\beta}$

	$\mathbf{v}$	$\boldsymbol{\xi}$
spatial transformation	$\mathbf{v}_2 = \mathbf{S}_{12}^T \mathbf{v}_1$	$\boldsymbol{\xi}_2 = \mathbf{S}_{12}^T \boldsymbol{\xi}_1$
implicit constraint eq.	$\mathbf{C}\mathbf{v} = \mathbf{0}$	$\mathbf{C}\boldsymbol{\xi} = \mathbf{0}$
explicit constraint eq.	$\mathbf{v} = \mathbf{H}\boldsymbol{\beta}$	$\boldsymbol{\xi} = \mathbf{H}\widehat{\boldsymbol{\xi}}$
joint-space counterpart	$\boldsymbol{\beta}$	$\widehat{\boldsymbol{\xi}}$

. . .

Let us now investigate algebraic constraints imposed on dynamic and adjoint systems: (3.14) and (4.14), respectively. Table 4.2 presents the detailed comparison between these quantities. Specifically, one can see that the constraints imposed on adjoint variables are shifted towards the higher-order derivative compared to the geometric constraints formulated for joints in a multibody system. Consequently, index-1 formulation of the adjoint system (4.11) involves jerk-level constraints instead of velocity-level constraints, which is the case for the underlying EOM (3.14) [1].

[1] Let us remark that constrained Hamilton's canonical equations form a system of DAEs with a differential index reduced with respect to the acceleration-based formulations (e.g. Newton-Euler or Lagrange's). This fact is one of the virtues of the Hamilton's formulation.

Table 4.2: Analogies between different algebraic constraints imposed on the multibody ($\boldsymbol{\Phi} = \mathbf{0}$) and the adjoint systems ($\boldsymbol{\Psi} = \mathbf{0}$)

	$\boldsymbol{\Phi}(\mathbf{q})$	$\boldsymbol{\Psi}(\mathbf{q}, \boldsymbol{\xi})$
position-level constr.	$\boldsymbol{\Phi} = \mathbf{0}$ (3.1)	$\times$
velocity-level constr.	$\mathbf{C}\mathbf{v} = \mathbf{0}$ (3.13)	$\mathbf{C}\boldsymbol{\xi} = \mathbf{0}$ (4.27)
acceleration-level constr.	$\mathbf{C}\dot{\mathbf{v}} = -\dot{\mathbf{C}}\mathbf{v}$ (3.10)	$\mathbf{C}\boldsymbol{\eta} = -\dot{\mathbf{C}}\boldsymbol{\xi}$ (4.10b)
jerk-level constr.	n.a.	$\mathbf{C}\dot{\boldsymbol{\eta}} = -\mathbf{r}_c$ (4.11b)

Equations appearing in tab. 4.2 represent implicit characterization of the constraints [43]. An equivalent form can be obtained with the aid of the motion subspace $\mathbf{H}$ that specifies the allowable velocity range space in $\mathcal{R}^6$ [64]. For example, eqs. (4.26) describe explicit velocity-level and acceleration-level constraints of the adjoint system, respectively. Subsequently, the differentiation of eq. (4.26b) yields the explicit form of the jerk-level constraints:

$$\dot{\boldsymbol{\eta}} = \mathbf{H}\dot{\widehat{\boldsymbol{\eta}}} + 2\dot{\mathbf{H}}\widehat{\boldsymbol{\eta}} + \ddot{\mathbf{H}}\widehat{\boldsymbol{\xi}}. \tag{4.28}$$

Once we plug eq. (4.28) into the adjoint system (4.11b), it is possible to premultiply both sides of the equation with the term $\mathbf{H}^T$ to come up with the joint-space adjoint equations:

$$\dot{\widehat{\boldsymbol{\xi}}} = \widehat{\boldsymbol{\eta}}, \tag{4.29a}$$

$$\left(\mathbf{H}^T\mathbf{M}\mathbf{H}\right)\dot{\widehat{\boldsymbol{\eta}}} = -\underbrace{\mathbf{H}^T\left(\mathbf{r} + 2\mathbf{M}\dot{\mathbf{H}}\widehat{\boldsymbol{\eta}} + \mathbf{M}\ddot{\mathbf{H}}\widehat{\boldsymbol{\xi}}\right)}_{\widehat{\mathbf{r}}} \quad \Rightarrow \quad \widehat{\mathbf{M}}\dot{\widehat{\boldsymbol{\eta}}} = -\widehat{\mathbf{r}}. \tag{4.29b}$$

Equation (4.29) may be solved backwards in time for unknown variables $\widehat{\boldsymbol{\eta}}$ and $\widehat{\boldsymbol{\xi}}$. The initial values are obtained by projecting boundary conditions in absolute coordinates (4.13) onto the subspace $\mathbf{H}$:

$$\left(\mathbf{H}^T\mathbf{H}\right)\widehat{\boldsymbol{\xi}}_0 = \mathbf{H}^T\boldsymbol{\xi}_0, \quad \left(\mathbf{H}^T\mathbf{H}\right)\widehat{\boldsymbol{\eta}}_0 = \mathbf{H}^T\left(\boldsymbol{\eta}_0 - \dot{\mathbf{H}}\widehat{\boldsymbol{\xi}}_0\right). \tag{4.30}$$

Please note that the matrix $\mathbf{H}^T\mathbf{H}$ is invertible in non-singular configurations. Let us also pinpoint that the RHS of eq. (4.29b) depends on absolute coordinates, i.e. $\widehat{\mathbf{r}} = \widehat{\mathbf{r}}(\mathbf{q}, \mathbf{v}, \dot{\mathbf{v}}, \boldsymbol{\sigma}, \boldsymbol{\lambda})$; however, the components that appear underneath are straightforward and relatively easy to formulate algorithmically. Moreover, eqs. (4.29) constitute ordinary differential equations since the constraints are enforced explicitly via eq. (4.28), and no additional algebraic constraints must be taken into account. Once the adjoint variables $\widehat{\boldsymbol{\eta}}, \widehat{\boldsymbol{\xi}}$ are evaluated for the entire time domain, their absolute coordinate counterparts can

be calculated via eqs. (4.26). Ultimately, they can be utilized to efficiently compute the gradient of a performance measure (4.15).

4.4 A commutative property of the adjoint method

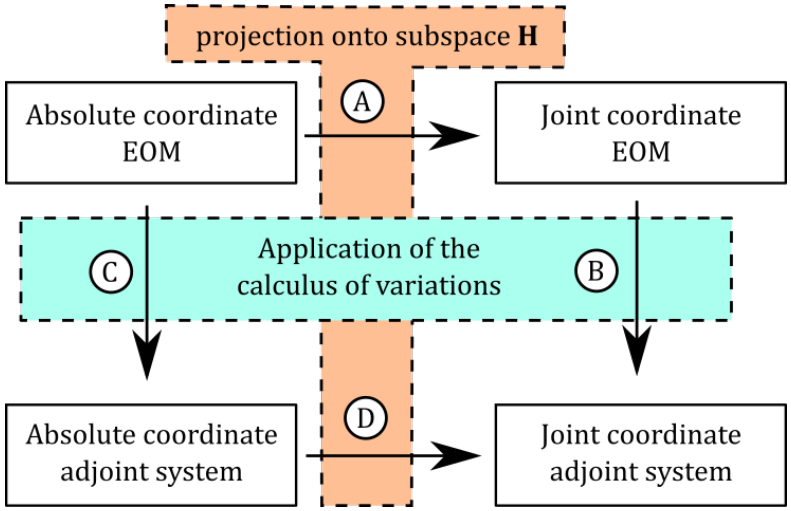


Figure 4.1: Possible paths to obtain adjoint system (4.29).

Section 4.3 used projection methods to develop independent adjoint variables, which allowed to write joint-space adjoint equations as a set of ODEs. This is equivalent to the path marked as CD in figure 4.1. Here, we briefly show that the same outcome can be obtained via path AB, i.e. by applying variational calculus to the EOM expressed in joint-space. Without loss of generality, we can assume $\boldsymbol{\beta} := \dot{\boldsymbol{\alpha}}$ to make the calculations simpler. First, let us rewrite eq. (4.25) and expand the products $\widehat{\boldsymbol{\eta}}^T\boldsymbol{\varphi}$, $\widehat{\boldsymbol{\xi}}^T\dot{\boldsymbol{\varphi}}$ into the following form:

$$\overline{J} = \int_0^{t_f} \left[h + \widehat{\boldsymbol{\eta}}^T(\widehat{\mathbf{p}} - \widehat{\mathbf{M}}\dot{\boldsymbol{\alpha}}) + \widehat{\boldsymbol{\xi}}^T(\dot{\widehat{\mathbf{p}}} - \mathbf{H}^T\mathbf{f} - \dot{\mathbf{H}}^T\mathbf{M}\mathbf{H}\dot{\boldsymbol{\alpha}}) \right] \mathrm{d}t + S. \tag{4.31}$$

Furthermore, we substitute $\boldsymbol{\tau} := \mathbf{H}^T\mathbf{f} + \dot{\mathbf{H}}^T\mathbf{M}\mathbf{H}\dot{\boldsymbol{\alpha}}$ in eq. (4.31) to be more terse in the remainder of this section. The variation of the performance measure (4.31) reads as:

$$\begin{aligned} \delta\overline{J} = \int_0^{t_f} \Big[&\big(h_{\mathbf{b}} - \widehat{\boldsymbol{\eta}}^T(\widehat{\mathbf{M}}\dot{\boldsymbol{\alpha}})_{\mathbf{b}} - \widehat{\boldsymbol{\xi}}^T\boldsymbol{\tau}_{\mathbf{b}}\big)\delta\mathbf{b} + \widehat{\boldsymbol{\eta}}^T\big(\delta\widehat{\mathbf{p}} - \widehat{\mathbf{M}}\delta\dot{\boldsymbol{\alpha}} - (\widehat{\mathbf{M}}\dot{\boldsymbol{\alpha}})_{\boldsymbol{\alpha}}\delta\boldsymbol{\alpha}\big) + \\ &\widehat{\boldsymbol{\xi}}^T(\delta\dot{\widehat{\mathbf{p}}} - \boldsymbol{\tau}_{\boldsymbol{\alpha}}\delta\boldsymbol{\alpha} - \boldsymbol{\tau}_{\dot{\boldsymbol{\alpha}}}\delta\dot{\boldsymbol{\alpha}}) + h_{\boldsymbol{\alpha}}\delta\boldsymbol{\alpha} + h_{\dot{\boldsymbol{\alpha}}}\delta\dot{\boldsymbol{\alpha}} \Big] \mathrm{d}t + S_{\boldsymbol{\alpha}}\delta\boldsymbol{\alpha}|_{t_f} + S_{\dot{\boldsymbol{\alpha}}}\delta\dot{\boldsymbol{\alpha}}|_{t_f}\,, \end{aligned} \tag{4.32}$$

which can be integrated by parts (c.f. eq. (4.6)) and reordered to come up with the following equation:

$$\begin{aligned}\delta\overline{J} = \int_0^{t_f} \Big[\Big(h_{\mathbf{b}} - \widehat{\boldsymbol{\eta}}^T(\widehat{\mathbf{M}}\dot{\boldsymbol{\alpha}})_{\mathbf{b}} - \widehat{\boldsymbol{\xi}}^T\boldsymbol{\tau}_{\mathbf{b}}\Big)\delta\mathbf{b} + \Big(\widehat{\boldsymbol{\eta}}^T - \dot{\widehat{\boldsymbol{\xi}}}^T\Big)\delta\widehat{\mathbf{p}} \;\; + (h_{\boldsymbol{\alpha}} - \dot{h}_{\dot{\boldsymbol{\alpha}}})\delta\boldsymbol{\alpha} + \\ \Big(\dot{\widehat{\boldsymbol{\eta}}}^T\widehat{\mathbf{M}} + \mathbf{e}^T\big(\dot{\widehat{\mathbf{M}}} - (\widehat{\mathbf{M}}\dot{\boldsymbol{\alpha}})_{\boldsymbol{\alpha}}\big) + \widehat{\boldsymbol{\xi}}^T\big(\dot{\boldsymbol{\tau}}_{\dot{\boldsymbol{\alpha}}} - \boldsymbol{\tau}_{\boldsymbol{\alpha}}\big) + \dot{\widehat{\boldsymbol{\xi}}}^T\boldsymbol{\tau}_{\dot{\boldsymbol{\alpha}}}\Big)\delta\boldsymbol{\alpha}\Big]\,\mathrm{d}t + \\ \widehat{\boldsymbol{\xi}}^T\delta\widehat{\mathbf{p}}\Big|_{t_f} - \widehat{\boldsymbol{\eta}}^T\widehat{\mathbf{M}}\delta\boldsymbol{\alpha}\Big|_{t_f} - \widehat{\boldsymbol{\xi}}^T\boldsymbol{\tau}_{\dot{\boldsymbol{\alpha}}}\delta\boldsymbol{\alpha}\Big|_{t_f} + (h_{\dot{\boldsymbol{\alpha}}} + S_{\boldsymbol{\alpha}})\delta\boldsymbol{\alpha}|_{t_f} + S_{\dot{\boldsymbol{\alpha}}}\delta\dot{\boldsymbol{\alpha}}|_{t_f}\,.\end{aligned} \tag{4.33}$$

At this point, we follow the same principle as in section 4.3. The goal is to express the variation of the cost functional in terms of the variation of the design parameters. Hence, the expressions that stand next to the state variations $\delta\widehat{\mathbf{p}}, \delta\boldsymbol{\alpha}$ must be equated to zero. The resultant equations are analogous to eqs. (4.29), i.e.:

$$\dot{\widehat{\boldsymbol{\xi}}} = \widehat{\boldsymbol{\eta}}, \qquad \widehat{\mathbf{M}}\dot{\widehat{\boldsymbol{\eta}}} = -\widehat{\mathbf{r}}^*. \tag{4.34}$$

On the other hand, the RHS of the eq (4.34) is calculated with respect to a different set of state variables. It can be written implicitly as: $\widehat{\mathbf{r}}^* = \widehat{\mathbf{r}}^*(\boldsymbol{\alpha}, \dot{\boldsymbol{\alpha}}, \ddot{\boldsymbol{\alpha}})$. Consequently, by invoking the property $\dot{\widehat{\boldsymbol{\xi}}} = \widehat{\boldsymbol{\eta}}$, right-hand side has the following form:

$$\widehat{\mathbf{r}}^* = h_{\boldsymbol{\alpha}} - \frac{\mathrm{d}}{\mathrm{d}t}(h_{\dot{\boldsymbol{\alpha}}}) + \big(\dot{\widehat{\mathbf{M}}} - (\widehat{\mathbf{M}}\dot{\boldsymbol{\alpha}})_{\boldsymbol{\alpha}} + \boldsymbol{\tau}_{\dot{\boldsymbol{\alpha}}}\big)^T\mathbf{e} + \big(\dot{\boldsymbol{\tau}}_{\dot{\boldsymbol{\alpha}}} - \boldsymbol{\tau}_{\boldsymbol{\alpha}}\big)^T\widehat{\boldsymbol{\xi}}. \tag{4.35}$$

The derivatives that appear in eq. (4.35) are much more involved when compared with those arising in eq. (4.29) or (B.7). Moreover, a vast majority of expressions in eq. (B.7) are readily available for implementation in closed form [56, 108]. On the other hand, formulation (4.34) does not involve constraint Jacobians nor multipliers associated with the constraints, such as $\boldsymbol{\sigma}$ or $\boldsymbol{\lambda}$.

As was the case in section 4.2, equation (4.33) does not provide consistent boundary conditions for the terminal cost $S(\boldsymbol{\alpha}, \dot{\boldsymbol{\alpha}})$. As a remedy, we invoke eq. (3.27a) at t_f pre-multiplied by a terminal-time adjoint variable $\boldsymbol{\nu} \in \mathcal{R}^{n_\beta}$ and calculate its total variation. The resultant expression can be written as:

$$\delta\boldsymbol{\nu}^T\big(\widehat{\mathbf{p}} - \widehat{\mathbf{M}}\dot{\boldsymbol{\alpha}}\big)\Big|_{t_f} = \boldsymbol{\nu}^T\delta\widehat{\mathbf{p}}\Big|_{t_f} - \boldsymbol{\nu}^T\widehat{\mathbf{M}}\delta\dot{\boldsymbol{\alpha}}\Big|_{t_f} - \boldsymbol{\nu}^T(\widehat{\mathbf{M}}\dot{\boldsymbol{\alpha}})_{\boldsymbol{\alpha}}\delta\boldsymbol{\alpha}\Big|_{t_f}. \tag{4.36}$$

By appending equation (4.36) to the variation of the performance measure, we come up with the following set of boundary conditions for the adjoint variables:

$$\widehat{\mathbf{M}}\boldsymbol{\nu}\Big|_{t_f} = S_{\dot{\boldsymbol{\alpha}}}^T \tag{4.37a}$$

$$\widehat{\boldsymbol{\xi}}\Big|_{t_f} = -\boldsymbol{\nu} \tag{4.37b}$$

$$\widehat{\mathbf{M}}\widehat{\boldsymbol{\eta}}\Big|_{t_f} = S_{\boldsymbol{\alpha}}^T + h_{\dot{\boldsymbol{\alpha}}}^T\Big|_{t_f} + \left((\widehat{\mathbf{M}}\dot{\boldsymbol{\alpha}})_{\boldsymbol{\alpha}} - \boldsymbol{\tau}_{\dot{\boldsymbol{\alpha}}}\right)^T \widehat{\boldsymbol{\xi}}\Big|_{t_f} \tag{4.37c}$$

Ultimately, the gradient of the performance measure expressed via joint-space costate variables has the following form:

$$(\nabla_{\mathbf{b}} J)^T = h_{\mathbf{b}}^T - (\widehat{\mathbf{M}}\dot{\boldsymbol{\alpha}})_{\mathbf{b}}^T \widehat{\boldsymbol{\eta}} - \boldsymbol{\tau}_{\mathbf{b}}^T \widehat{\boldsymbol{\xi}} \tag{4.38}$$

This section revealed an interesting symmetry arising on the junction of the variational calculus and projection methods. We showed that the adjoint system defined in the joint coordinate space has a unique form regardless of the order of transformations applied to the underlying equations of motion. This property is visualized in figure 4.1. The variational manipulation and projection onto the subspace $\mathbf{H}$ are commutative when applied to eq. (3.14).

The gradient of the performance measure can be computed via different sets of the adjoint variables that may be expressed as absolute coordinates $\boldsymbol{\eta}, \boldsymbol{\xi}, \boldsymbol{\mu}$ or defined in joint-space: $\widehat{\boldsymbol{\eta}}, \widehat{\boldsymbol{\xi}}$. Both sets maintain certain virtues and disadvantages; however, they can be easily mapped from one set to another. The accompanying software implementation developed in the course of the work relies on the gradient evaluated with the aid of eq. (4.15). The approach employing equation (4.38) presents a possible road for future investigations.

Last, but not least, the divide-and-conquer approach relies heavily on the absolute coordinate variables while treating their joint-space counterparts as the primary (integration) variables. In the following section, we will show that by adequate combination of both sets of variables it is possible to establish the DCA scheme for the adjoint system (4.14).

4.5 Hamiltonian Adjoint-based DCA

This section outlines the necessary steps to apply the recursive divide-and-conquer pattern established in section 3.4.1 to the adjoint system (4.14). Practically hardly any changes must be applied to the DCA pattern since the leading matrix is the same for all considered systems (3.14), (3.15), (4.14), and only expressions that evaluate the right-

Table 4.3: Two analogous representations of the algebraic constraints at different levels. Underlined elements are assembly and disassembly bias terms.

constraints:	implicit (stacked)	explicit (body-wise)
velocity eqs.	$\dot{\mathbf{\Phi}} = \mathbf{C}\mathbf{v} = \mathbf{0}$	$\mathbf{V}_1^B = \mathbf{V}_2^A + \mathbf{H}_j\boldsymbol{\beta}_j$
acceleration eqs.	$\ddot{\mathbf{\Phi}} = \mathbf{C}\dot{\mathbf{v}} - \boldsymbol{\gamma} = \mathbf{0}$	$\dot{\mathbf{V}}_1^B = \dot{\mathbf{V}}_2^A + \mathbf{H}_j\dot{\boldsymbol{\beta}}_j + \underline{\dot{\mathbf{H}}_j\boldsymbol{\beta}_j}$
adjoint eqs.	$\ddot{\mathbf{\Psi}} = \mathbf{C}\dot{\boldsymbol{\eta}} + s \cdot \mathbf{r}_c = \mathbf{0}$	$\dot{\boldsymbol{\eta}}_1^B = \dot{\boldsymbol{\eta}}_2^A + \mathbf{H}_j\dot{\widehat{\boldsymbol{\eta}}}_j + \underline{s \cdot \left(2\dot{\mathbf{H}}_j\widehat{\boldsymbol{\eta}}_j + \ddot{\mathbf{H}}_j\widehat{\boldsymbol{\xi}}_j\right)}$

hand side terms must be altered. Specifically, two key components must be amended: the assembly/disassembly *bias term* (equal to zero in the case of velocity-level EOM) and initial setup of system coefficients $\boldsymbol{\chi}_{10}^{A,B} - \boldsymbol{\chi}_{20}^{A,B}$, ${}^{acc}\boldsymbol{\zeta}_{10}^{A,B} - {}^{acc}\boldsymbol{\zeta}_{20}^{A,B}$. We will show that the former component is intimately connected with algebraic constraints of the DAE system. Since velocity-level equation (3.14a), acceleration-level equation (3.15), and adjoint system (4.14b) involve constraints whose "differential level" rises incrementally (c.f. tab 4.2), here we additionally take into account the acceleration-level system as an intermediate link between equations (3.14a) and (4.14b).

To this end, let us rewrite time derivative of eq. (4.10b), which defines algebraic constraints of eq. (4.11b):

$$\ddot{\mathbf{\Psi}} = \mathbf{C}\dot{\boldsymbol{\eta}} + 2\dot{\mathbf{C}}\boldsymbol{\eta} + \ddot{\mathbf{C}}\boldsymbol{\xi} = \mathbf{C}\dot{\boldsymbol{\eta}} + \mathbf{r}_c = \mathbf{0} \tag{4.39}$$

As explained initially in section 4.4, relations (4.39) and (4.28) describe the same constraint equation; however, the former is defined implicitly, whereas the latter is explicit. Note that one equation employs constraint Jacobian matrix $\mathbf{C}$ and its derivatives, while the other involves the null space of Jacobian, i.e. the allowable motion subspace $\mathbf{H}$. Moreover, these analogies hold at other levels of the constraints, specifically at velocity and acceleration levels. Table 4.3 presents both representations for all considered levels. The symbol $s = -1$ signifies reversal of the sign due to the numerical integration done backwards in time. We aim to utilize the rightmost column of tab. 4.3 in further derivations. For example, explicit constraints of the velocity level equations have already been employed in equation (3.36).

Let us now rewrite eq. (3.36) in a more general form, which allows us to include higher level constraints:

$$\mathbf{D}_j^T(\dot{\boldsymbol{\eta}}_1^B - \dot{\boldsymbol{\eta}}_2^A - \boldsymbol{\zeta}_{\text{bias}}) = \mathbf{0}, \tag{4.40}$$

where $\dot{\boldsymbol{\eta}}_1^B, \dot{\boldsymbol{\eta}}_2^A$ can be substituted by $\dot{\mathbf{V}}_1^B, \dot{\mathbf{V}}_2^A$ or $\mathbf{V}_1^B, \mathbf{V}_2^A$ depending on the considered problem. This alteration affects equations (3.37), (3.38); however, by updating the auxiliary variable $\mathbf{c}_j$ in the following way: $\mathbf{c}_j = \mathbf{W}_j(\boldsymbol{\zeta}_{10}^B - \boldsymbol{\zeta}_{20}^A - \boldsymbol{\zeta}_{\text{bias}})$ we achieve the same structure

of the remaining equations (3.39–3.42). The exact form of the bias term $\boldsymbol{\zeta}_{\text{bias}}$ depends on the solved system of equations; for example, the velocity problem presented in sec. 3.4.1 uses $\boldsymbol{\zeta}_{\text{bias}} = \mathbf{0}$, while the adjoint system involves $\boldsymbol{\zeta}_{\text{bias}} = s \cdot \left(2\dot{\mathbf{H}}_j \widehat{\boldsymbol{\eta}}_j + \ddot{\mathbf{H}}_j \widehat{\boldsymbol{\xi}}_j\right)$. Let us pinpoint that the specified amendment must be applied each time a compound bodies get assembled or disassembled; nevertheless, the main loop of the algorithm remains intact.

The second component that needs to be addressed refers to the initial setup of the coefficients appearing in the equations. Let us establish the relations between two neighboring (physical or articulated) bodies A and B for both adjoint and acceleration systems:

$$\dot{\boldsymbol{\eta}}_1^A = \boldsymbol{\zeta}_{11}^A \mathbf{K}_1^A + \boldsymbol{\zeta}_{12}^A \mathbf{K}_2^A + \boldsymbol{\chi}_{10}^A, \tag{4.41}$$

$$\dot{\boldsymbol{\eta}}_2^A = \boldsymbol{\zeta}_{21}^A \mathbf{K}_1^A + \boldsymbol{\zeta}_{22}^A \mathbf{K}_2^A + \boldsymbol{\chi}_{20}^A, \tag{4.42}$$

$$\dot{\boldsymbol{\eta}}_1^B = \boldsymbol{\zeta}_{11}^B \mathbf{K}_1^B + \boldsymbol{\zeta}_{12}^B \mathbf{K}_2^B + \boldsymbol{\chi}_{10}^B, \tag{4.43}$$

$$\dot{\boldsymbol{\eta}}_2^B = \boldsymbol{\zeta}_{21}^B \mathbf{K}_1^B + \boldsymbol{\zeta}_{22}^B \mathbf{K}_2^B + \boldsymbol{\chi}_{20}^B, \tag{4.44}$$

$$\dot{\mathbf{V}}_1^A = \boldsymbol{\zeta}_{11}^A \mathbf{L}_1^A + \boldsymbol{\zeta}_{12}^A \mathbf{L}_2^A + {}^{acc}\boldsymbol{\zeta}_{10}^A, \tag{4.45}$$

$$\dot{\mathbf{V}}_2^A = \boldsymbol{\zeta}_{21}^A \mathbf{L}_1^A + \boldsymbol{\zeta}_{22}^A \mathbf{L}_2^A + {}^{acc}\boldsymbol{\zeta}_{20}^A, \tag{4.46}$$

$$\dot{\mathbf{V}}_1^B = \boldsymbol{\zeta}_{11}^B \mathbf{L}_1^B + \boldsymbol{\zeta}_{12}^B \mathbf{L}_2^B + {}^{acc}\boldsymbol{\zeta}_{10}^B, \tag{4.47}$$

$$\dot{\mathbf{V}}_2^B = \boldsymbol{\zeta}_{21}^B \mathbf{L}_1^B + \boldsymbol{\zeta}_{22}^B \mathbf{L}_2^B + {}^{acc}\boldsymbol{\zeta}_{20}^B. \tag{4.48}$$

One can see that equations (4.41–4.48) are analogous to the set (3.30–3.33). The quantities $\mathbf{K}_{1,2}^{A,B}, \mathbf{L}_{1,2}^{A,B} \in \mathcal{R}^6$ are tied with Lagrange multipliers $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$, respectively. For example, $\mathbf{K}_1^A = \mathbf{D}_i \boldsymbol{\mu}_i$ and $\mathbf{L}_1^B = \mathbf{D}_j \boldsymbol{\lambda}_j$ (c.f. sec. 3.4.1). Symbols $\dot{\boldsymbol{\eta}}_{1,2}^{A,B} \in \mathcal{R}^6$ that involve both subscript and superscript refer to the particular spatial vector representation of the quantity $\dot{\boldsymbol{\eta}}$ (it is a subset of the global stacked vector).

Since the leading matrices in the global formulation of all considered systems (3.14), (3.15), (4.14) are equal, coefficients $\boldsymbol{\zeta}_{11}^A - \boldsymbol{\zeta}_{22}^B$ remain the same between these systems at a given time instant. Moreover, during the assembly and disassembly phases equations (3.42) and (3.39) remain valid, and no amendment is necessary. Conversely, the initial setup of the parameters $\boldsymbol{\chi}_{10}^{A,B} - \boldsymbol{\chi}_{20}^{A,B}$, ${}^{acc}\boldsymbol{\zeta}_{10}^{A,B} - {}^{acc}\boldsymbol{\zeta}_{20}^{A,B}$ does need to be altered in order to reflect the appropriate right-hand side of the underlying equation. Given all required inputs, such as: kinematic parameters, reactions, design parameters, or control signals, this can be done in parallel, where each thread iterates over a specified number of bodies. Example (4.1) presents detailed description on how these quantities can be evaluated.

Example 4.1. Calculate the leaf-body coefficients for velocity $(\boldsymbol{\zeta}_{10}^A, \boldsymbol{\zeta}_{20}^A)$, acceleration $({}^{acc}\boldsymbol{\zeta}_{10}^A, {}^{acc}\boldsymbol{\zeta}_{20}^A)$, and adjoint $(\boldsymbol{\chi}_{10}^A, \boldsymbol{\chi}_{20}^A)$ systems.

Coefficients of the velocity-level equation can be extracted by comparing equation (3.29) with eqs. (3.30), (3.31). It is worth noting that computing the value of $\boldsymbol{\zeta}_{20}^A$ requires additional premultiplication by the shift matrix $\mathbf{S}_{21}^A$:

$$\boldsymbol{\zeta}_{10}^{A} = \left(\mathbf{M}_1^A\right)^{-1}(\mathbf{H}_i\widehat{\mathbf{p}}_i - \mathbf{S}_{12}^A\mathbf{H}_j\widehat{\mathbf{p}}_j), \tag{4.49a}$$

$$\boldsymbol{\zeta}_{20}^{A} = \left(\mathbf{M}_2^A\right)^{-1}(\mathbf{S}_{21}^A\mathbf{H}_i\widehat{\mathbf{p}}_i - \mathbf{H}_j\widehat{\mathbf{p}}_j), \tag{4.49b}$$

$$\begin{aligned}\boldsymbol{\zeta}_{11}^{A} &= \left(\mathbf{M}_1^A\right)^{-1}, \quad \boldsymbol{\zeta}_{12}^{A} = \left(\mathbf{M}_1^A\right)^{-1}\mathbf{S}_{12}^A,\\ \boldsymbol{\zeta}_{22}^{A} &= \left(\mathbf{M}_2^A\right)^{-1}, \quad \boldsymbol{\zeta}_{21}^{A} = \left(\mathbf{M}_2^A\right)^{-1}\mathbf{S}_{21}^A.\end{aligned} \tag{4.49c}$$

For completeness, eq. (4.49c) presents main coefficients derived from relation (3.29). These quantities remain the same for all subproblems discussed in this example, i.e., $\boldsymbol{\zeta}_{ab}^{A} = {}^{acc}\boldsymbol{\zeta}_{ab}^{A} = \boldsymbol{\chi}_{ab}^{A}$ for $a, b = \{1, 2\}$.

Next, to compute coefficients of the acceleration-level equation, let us take the time derivative of eq. (3.29) (c.f. eq. (3.52)) and substitute left-hand side of eq. (3.53) with its RHS. We come up with the following result:

$$\dot{\mathbf{P}}_1^A = \mathbf{M}_1^A\dot{\mathbf{V}}_1^A + \dot{\mathbf{M}}_1^A\mathbf{V}_1^A = \mathbf{D}_i\dot{\boldsymbol{\sigma}}_i - \mathbf{S}_{12}^A\mathbf{D}_j\dot{\boldsymbol{\sigma}}_j + \mathbf{F}_1^A - \dot{\mathbf{S}}_{C1}^A\mathbf{P}_1^A,$$

which upon comparison with equations (4.45), (4.46) yields:

$${}^{acc}\boldsymbol{\zeta}_{10}^{A} = \left(\mathbf{M}_1^A\right)^{-1}(\mathbf{F}_1^A - \dot{\mathbf{S}}_{C1}^A\mathbf{P}_1^A - \dot{\mathbf{M}}_1^A\mathbf{V}_1^A), \tag{4.50a}$$

$${}^{acc}\boldsymbol{\zeta}_{20}^{A} = \left(\mathbf{M}_2^A\right)^{-1}(\mathbf{F}_2^A - \dot{\mathbf{S}}_{C2}^A\mathbf{P}_2^A - \dot{\mathbf{M}}_2^A\mathbf{V}_2^A). \tag{4.50b}$$

Finally, coefficients of the adjoint system are slightly more involved to obtain. First, let us take the time derivative of eq.(4.22b), that relates two spatial vectors $\dot{\boldsymbol{\eta}}_2^A$ and $\dot{\boldsymbol{\eta}}_1^A$ along a rigid body:

$$\dot{\boldsymbol{\eta}}_1^A = \mathbf{S}_{21}^T\dot{\boldsymbol{\eta}}_2^A + 2\dot{\mathbf{S}}_{21}^T\boldsymbol{\eta}_2^A + \ddot{\mathbf{S}}_{21}^T\boldsymbol{\xi}_2^A. \tag{4.51}$$

Let us note a trivial index shift performed ad hoc. Concurrently, the underlying adjoint equation can be established by rewriting eqs. (4.11b) and (B.7) in a body-wise format. By denoting $\boldsymbol{\mathcal{R}}_1^A$ as a body-wise counterpart of the global quantity $\mathbf{r}$ (computed at handle 1 of the body A), we come up with the following formula:

$$\mathbf{M}_1^A\dot{\boldsymbol{\eta}}_1^A = \mathbf{D}_i\dot{\boldsymbol{\mu}}_i - \mathbf{S}_{12}^A\mathbf{D}_j\dot{\boldsymbol{\mu}}_j + \boldsymbol{\mathcal{R}}_1^A. \tag{4.52}$$

The relation for $\boldsymbol{\chi}_{10}^A$ is straightforward; however, it is not necessarily the case for $\boldsymbol{\chi}_{20}^A$. To obtain the second coefficient one needs to premultiply eq. (4.52) by $\mathbf{S}_{21}^A$ (which was the case in both former examples), but also substitute eq. (4.51) into eq. (4.52). Ultimately, the following relations are obtained:

$$\boldsymbol{\chi}_{10}^{A} = \left(\mathbf{M}_{1}^{A}\right)^{-1} \boldsymbol{\mathcal{R}}_{1}^{A}, \tag{4.53a}$$

$$\boldsymbol{\chi}_{20}^{A} = \left(\mathbf{M}_{2}^{A}\right)^{-1} \mathbf{S}_{21}^{A} \left(\boldsymbol{\mathcal{R}}_{1}^{A} - s \cdot \mathbf{M}_{1}^{A} (2\dot{\mathbf{S}}_{21}^{T} \boldsymbol{\eta}_{2}^{A} + \ddot{\mathbf{S}}_{21}^{T} \boldsymbol{\xi}_{2}^{A}) \right). \tag{4.53b}$$

Alterations presented in this section allow us to strictly follow the methodology laid out in sec. 3.4.1. The quantities required to accurately evaluate the right-hand side term $\boldsymbol{\mathcal{R}}_{1}^{A}$ involve kinematic parameters $\mathbf{q}, \mathbf{v}, \dot{\mathbf{v}}$, reactions $\boldsymbol{\lambda}$, and design parameters or control signals $\mathbf{b}$. As far as HADCA algorithm is concerned, the integration of EOM forward in time yields joint-space variables, such as $\boldsymbol{\alpha}, \boldsymbol{\beta}, \dot{\boldsymbol{\beta}}$. Therefore, one may employ equation (3.21) and its derivative (c.f. tab. 4.3) to extract the absolute-coordinate counterparts needed for the adjoint problem. Similarly, the primary variables of the adjoint system's integration are joint-space variables $\widehat{\boldsymbol{\eta}}, \widehat{\boldsymbol{\xi}}$. The boundary values can be obtained by solving absolute-coordinate boundary system (4.13) and projecting the outcome on the allowable motion subspace $\mathbf{H}$ as in eq. (4.30). When the numerical integration routine demands it, the absolute adjoint variables $\boldsymbol{\eta}$ and $\boldsymbol{\xi}$ can be recreated from $\widehat{\boldsymbol{\eta}}, \widehat{\boldsymbol{\xi}}$ by combining equations (4.22) with eqs. (4.26). Specifically, they may be recursively applied along all bodies of the MBS in a similar fashion to velocity and acceleration (see tab. 4.3). Finally, once $\dot{\boldsymbol{\eta}}_{1}^{A}$ and $\dot{\boldsymbol{\eta}}_{2}^{A}$ are evaluated for all bodies via DCA scheme, $\dot{\widehat{\boldsymbol{\eta}}}$ and $\dot{\widehat{\boldsymbol{\xi}}}$ can be calculated by projecting explicit adjoint algebraic constraint onto the subspace $\mathbf{H}_{j}$:

$$\dot{\widehat{\boldsymbol{\eta}}}_{j} = \mathbf{H}_{j}^{T} \left(\dot{\boldsymbol{\eta}}_{1}^{B} - \dot{\boldsymbol{\eta}}_{2}^{A} - \boldsymbol{\zeta}_{\text{bias}} \right), \qquad \dot{\widehat{\boldsymbol{\xi}}}_{j} = s \cdot \widehat{\boldsymbol{\eta}}_{j} = -\widehat{\boldsymbol{\eta}}_{j}. \tag{4.54}$$

Consequently, each system of equations requires its own divide-and-conquer sweep. Altogether, there may appear up to four sweeps along the binary tree involving assembly and disassembly phases for each time instant, specifically:

1. velocity equations (3.14a) (figure 3.5) ,
2. force distribution (3.47) (figure 3.6) ,
3. acceleration equations (3.15) ,
4. adjoint system (4.14b) (figure 4.2) .

As stated in section 3.4.2, steps 1 and 2 can be combined for greater efficiency when the generalized force does not depend on velocity. Sweep 3 is necessary to extract $\dot{\boldsymbol{\beta}}$ and $\boldsymbol{\lambda}$ required to evaluate coefficients of the adjoint equations during sweep 4. The fact that coefficients associated with the leading matrix $\boldsymbol{\zeta}_{11}^{A} - \boldsymbol{\zeta}_{22}^{B}$ are equal for steps 1, 3, and 4

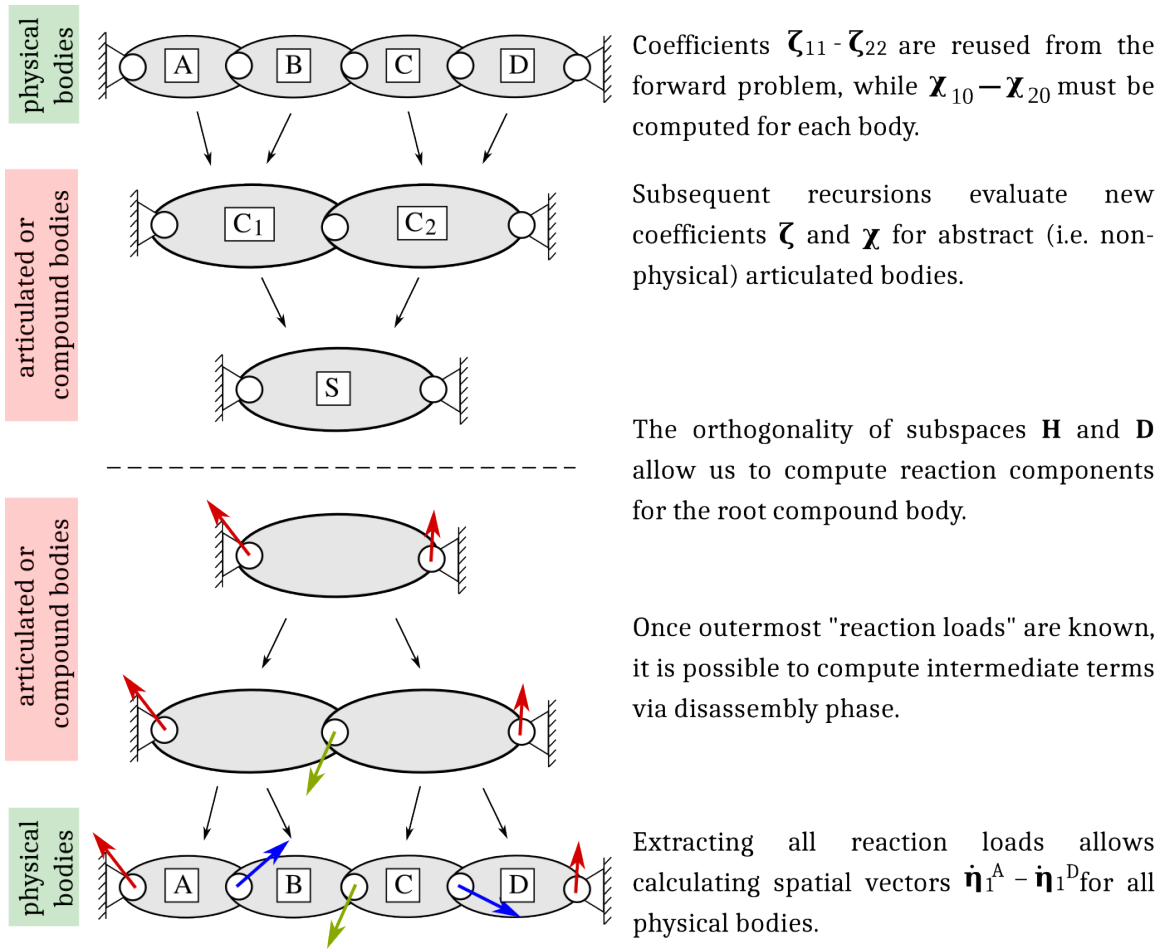


Figure 4.2: Stages of the DCA procedure for solving the adjoint system

can be exploited to increase the efficiency of the implementation. On the other hand, storing many dense matrices in memory would increase the memory footprint during the execution. Fortunately, the checkpointing approach described in section 4.2 provides a feasible way to overcome this issue.

4.6 Summary

This chapter looks into the adjoint sensitivity analysis. The ultimate goal is an efficient and reliable computation of the gradient of the performance measure. To this end, multiple formulations and approaches have been investigated herein. First, we come up with a detailed derivation of the adjoint system to the constrained Hamilton's equations of motion (3.14). Since the underlying EOM have a form of DAE, the resultant adjoint system consists of differential-algebraic equations as well.

In the following sections, we come up with various concepts allowing to solve the adjoint system more efficiently. Section 4.3 is dedicated to exposing certain rudimentary relationships, such as:

- Introduction of joint-space costate variables $\widehat{\boldsymbol{\eta}}, \widehat{\boldsymbol{\xi}}$
- Development of spatial relations between absolute adjoint variables (4.22)
- A bidirectional map between joint-space and absolute adjoint variables: eqs. (4.26) and (4.30)
- Analogies between physical and adjoint variables gathered in table 4.1

Discovered relations and analogies have shown that the same projector, which is utilized to project the redundant equations of motion onto the minimal joint's motion subspace $\mathbf{H}$, can be used to formulate the adjoint dynamics as a system of ordinary differential equations (4.29).

Subsequently, section 4.4 presents a discussion about two studied global formulations of the adjoint system: absolute (4.14) and formulated as ODEs (4.29). We were able to show that the latter system can be obtained from the former in two distinct ways. Although the resultant systems have the same form in both cases, they are described by different sets of variables, namely ODE system is based on absolute variables $\mathbf{q}, \mathbf{v}$, etc., whereas system (4.14) is described by joint variables $\boldsymbol{\alpha}, \dot{\boldsymbol{\alpha}}$, etc.

Section 4.5 is one of the most important in this thesis since it fulfills one of its primary goals by presenting a novel approach to the adjoint sensitivity analysis. It is devoted to the detailed derivation of divide-and-conquer relations for the adjoint system, which allows for parallelization of equation (4.14). Many relevant formulae for adjoint divide-and-conquer algorithm are identical to the equations describing the DCA of the forward problem; hence we present multiple references to section 3.4.1. Concurrently, the acceleration-level relationships for the DCA are delivered since they conveniently fill the gap arising between the velocity-level and adjoint-level relationships. The graphical presentation of the adjoint-level recursion based on the divide-and-conquer paradigm is presented in figure 4.2.

CHAPTER 5
NUMERICAL EXAMPLES

5.1 Introduction

This chapter presents the application results that demonstrate the validity of the adjoint methodology proposed in this work as well as encountered underlying challenges. To make the concepts presented in previous sections easier to grasp, the contents described here have been divided into subsections, where a different type of problem is solved, and the emphasis is put on a distinct aspect of the approach. The scope of this chapter encompasses the following problems:

- Section 5.2: optimal design of a planar multibody system and extensive comparison of the Hamiltonain-based adjoint approach with a variety of other methods, such as: finite differences, complex-step method, direct differentiation method, and other adjoint approaches.
- Section 5.3: optimal design of spatial pendulum, where the relations between joint-space and absolute adjoint variables derived in sec. 4.3 are numerically verified.
- Section 5.4: optimal control of planar open kinematic tree, where two different scenarios are taken into account and additional verification of the relationships between joint-space and absolute variables are shown.
- Section 5.5: optimal control of spatial closed-loop MBS with a mathematical model of DC motors.

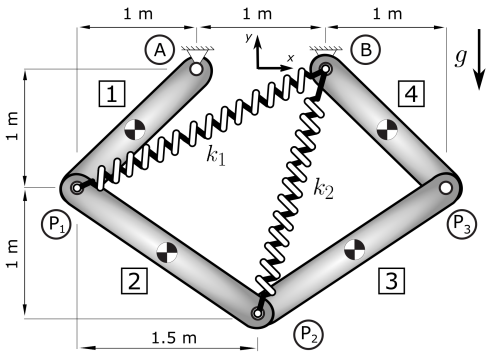


Figure 5.1: Five-bar planar mechanism

Table 5.1: Simulation parameters

Parameter	Value
Masses: m_1, m_4	1 kg
Masses: m_2, m_3	1.5 kg
Moment of inertia of $i-$th body about $z-$axis (l_i-length of $i-$th body)	$I_{zi} = \frac{1}{12} m_i l_i^2$ kg m^2
Stiffness of springs k_1, k_2	$100 \frac{N}{m}$
Free length of springs d_1^0, d_2^0	$[d_1^0, d_2^0]^T = [b_1, b_2]^T$
Final time t_f	5 s
Constant time step δt	0.005 s
Initial guess for the optimization procedure $\mathbf{b}_0$	$\mathbf{b}_0 = \left[\sqrt{5}, \sqrt{4.25}\right]^T$

5.2 Optimal design of a five-bar mechanism

In this section, we investigate a case of finding static equilibrium conditions of a five-bar mechanism placed in the gravitational field. This test example is commonly used by other authors as a benchmark for multibody optimization procedures [36, 31]. It can be easily solved classically via static equilibrium conditions; however, the goal is to test the validity and accuracy of the developed methods. The results of the adjoint sensitivity analysis will be compared against other approaches, such as direct differentiation and complex-step methods.

An initial configuration of the five-bar mechanism is presented in fig. 5.1. Two springs k_1 and k_2 are attached between the non-movable base body and bodies 1 and 2, respectively. The free lengths of the springs d_1^0, d_2^0 are adopted as design parameters $\mathbf{b} = [b_1, b_2]^T$, and their initial values correspond to a lack of pre-load in the springs. For example, vector $\mathbf{b} = [0, 0]^T$ would mean that the free lengths of springs are equal to zero. Some values of design variables (even non-zero positive ones) may produce ill-conditioned problems for which the dynamic response is oftentimes non-smooth. Since this leads to even worse conditioning of the adjoint system, bounds on the design variables have been imposed in the following form: $[0.9, 0.9]^T < \mathbf{b} < [7, 5]^T$. The parameters used in the simulation are gathered in tab. 5.1. The cost function which addresses the problem of finding static equilibrium has the following form:

$$J = \int_0^{t_f} \left(\mathbf{r}_{\mathbf{P}_2} - \mathbf{r}_{\mathbf{P}_2}^{\mathbf{init}}\right)^T \left(\mathbf{r}_{\mathbf{P}_2} - \mathbf{r}_{\mathbf{P}_2}^{\mathbf{init}}\right) dt, \tag{5.1}$$

where $\mathbf{r}_{\mathbf{P}_2}$ is a displacement of point $\mathbf{P}_2$ with respect to the origin of the global coordinate frame and $\mathbf{r}_{\mathbf{P}_2}^{\mathbf{init}}$ is its initial position. Only absolute coordinates are used to describe the configuration of the MBS in this example. The free lengths of the springs can be obtained analytically by formulating the conditions for static equilibrium. The minimal value of the

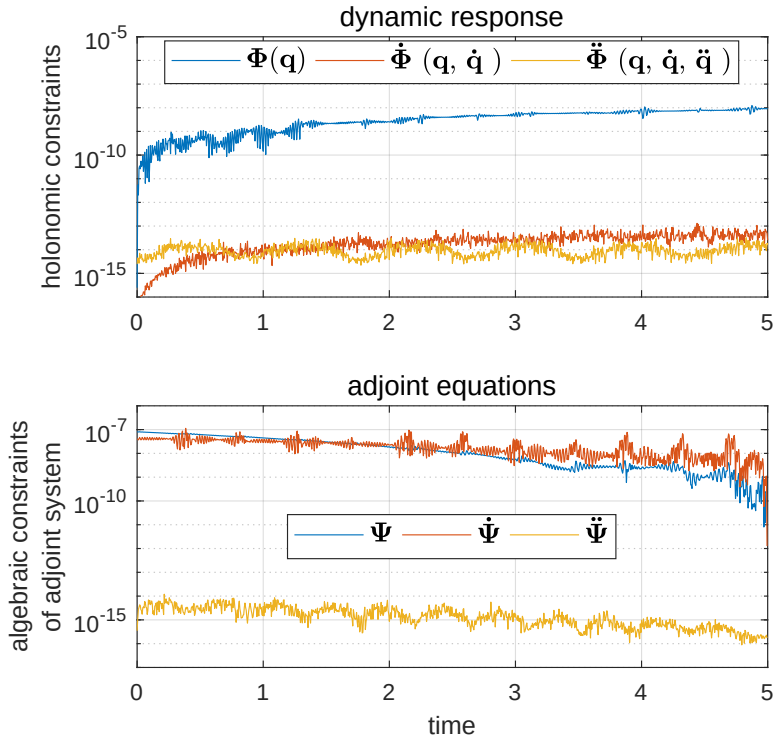


Figure 5.2: Constraint violation errors for holonomic and adjoint-based conditions

functional corresponding to such an analytical solution should be zero. This observation can be used to validate the calculations performed by the developed methods.

Moreover, the problem formulation in the form of the cost (5.1) assumes that $h = h(\mathbf{q}), S \equiv 0$ (c.f. eq. (4.1)). Reference [79] covers a similar test case formulated with respect to the magnitude of the reaction forces, i.e. $h = h(\boldsymbol{\lambda})$. The paper additionally explains how to handle additional intricate dependencies, such as acceleration $\ddot{\mathbf{q}}$ and $\boldsymbol{\lambda}$. In this example, the boundary conditions (4.13), as well as the derivatives of the adjoint RHS, simplify significantly due to the lack of dependency of performance measure neither on $\boldsymbol{\lambda}$ nor on $\ddot{\mathbf{q}}$.

Equations of motion (3.9) have been discretized and integrated forward in time starting from prescribed initial conditions. This action is inevitable to evaluate the cost functional (4.1), but it also yields a history of state variables which are utilized in eq. (4.8) for the backward solution of the adjoint system (4.14). A commercial code was used to integrate the differential equations, i.e. `RKF45` procedure provided by *Matlab*.

The top plot in fig. 5.2 shows the time-histories of position-level $\boldsymbol{\Phi}$, velocity-level $\dot{\boldsymbol{\Phi}}$, and acceleration-level algebraic constraints $\ddot{\boldsymbol{\Phi}}$ (see Eqs. (3.1), (3.2)). The curves at the bottom demonstrate the adjoint algebraic constraint equation $\boldsymbol{\Psi} = \boldsymbol{\Phi}_{\mathbf{q}}\boldsymbol{\xi}$ (see. eq. (4.27)), the time derivative $\dot{\boldsymbol{\Psi}} = \boldsymbol{\Phi}_{\mathbf{q}}\boldsymbol{\eta} + \dot{\boldsymbol{\Phi}}_{\mathbf{q}}\boldsymbol{\xi}$ (see eq. (4.10b)), and the second time derivative

$\ddot{\boldsymbol{\Psi}}$ (c.f. eq. (4.39) and tab. 4.2) that arise in the system of adjoint equations. The time-histories are recorded for the initial design $\mathbf{b}_0$. The forward dynamics problem formulated here as a system of DAEs is explicitly solved with the use of velocity-level constraint equations. Since the constraints are not artificially stabilized, a minor drift-off effect occurs, and the constraints $\dot{\boldsymbol{\Phi}}$ are fulfilled between 10^{-16} and 10^{-13} (m) throughout the simulation time.

There are some constraint violation errors at the position level, though. The trend is clearly seen in fig. 5.2. Fortunately, the errors are not too extensive and achieve the order of 10^{-8} $(\frac{\mathrm{m}}{\mathrm{s}})$ at the most. The same qualitative picture is observed when the adjoint constraints are taken into account. The conditions $\ddot{\boldsymbol{\Psi}} = \mathbf{0}$ are fulfilled up to the machine accuracy. There is some drift observed in the time-plots of $\boldsymbol{\Psi}$ and $\dot{\boldsymbol{\Psi}}$; however, the norms $||\boldsymbol{\Psi}||_2$, $||\dot{\boldsymbol{\Psi}}||_2$ do not exceed the value of 10^{-7}. The proposed approach does not require any constraint stabilization procedure to be involved in the solution process. The observed drifts in the algebraic constraints do not extensively affect the accuracy and stability of the performed simulations. This behavior comes from the stabilizing properties of constrained Hamilton's equations of motion. The underlying system of DAEs (cf. eq. (3.9)) is characterized by a reduced differential index [17].

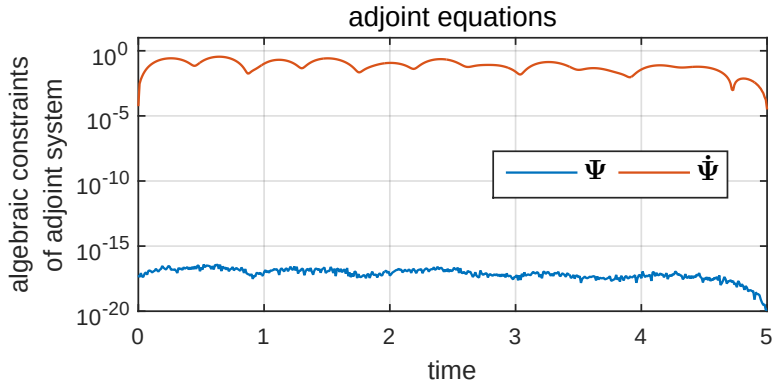


Figure 5.3: Constraint violation errors for adjoint-based system formulated as in refs. [77, 90]

The adjoint constraints presented in fig. 5.2 have been additionally compared with the constraints enclosed in a more classical, acceleration-based formulation (see fig. 5.3), in which the system of equations of motion is expressed as an index-3 DAE. [77, 90]. Since the dynamic equations of the classical formulation enforce the position-level constraints explicitly, one can expect that the derived adjoint system will accurately fulfill the constraints $\boldsymbol{\Psi} = \mathbf{0}$. On the other hand, this system is characterized by a higher differential index as compared to the case of Hamiltonian adjoint method which is more challenging

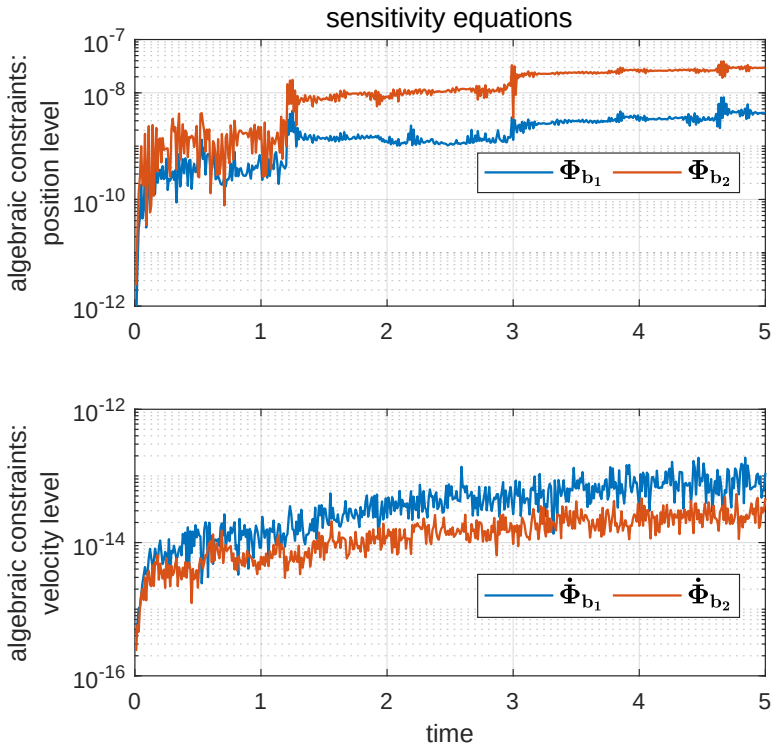


Figure 5.4: Constraint violation errors for sensitivity equations

to integrate. It was also observed, that the constraints associated with first and second time-derivatives of $\mathbf{\Psi}$ indicate higher violations as compared to the method proposed here.

Additionally, this example has been solved by means of the direct differentiation method, which was briefly explained in sec. 2.4.1 and has been thoroughly presented in refs. [78, 79]. The sensitivity equations obtained by direct differentiation of EOM (3.9) or (3.14) with respect to the vector $\mathbf{b}$ must be integrated forward in time along with the dynamic response evaluation. Once the sensitivity equations are solved, the derivative $\mathbf{q_b}$ can be inserted into eq. (2.5) in order to evaluate the gradient numerically. The explicit derivative corresponding to $\mathbf{q_b}$ reads as follows: $h_{\mathbf{q}} = 2\left(\mathbf{r}_{P_2} - \mathbf{r}_{P_2}^{init}\right)^T$. Figure 5.4 presents Euclidean norm of algebraic constraints associated with the sensitivity equations at the position $\mathbf{\Phi_b} = \mathbf{\Phi_q q_b}$ and velocity $\dot{\mathbf{\Phi}}_\mathbf{b} = \mathbf{\Phi_q}\dot{\mathbf{q}}_\mathbf{b} + \dot{\mathbf{\Phi}}_\mathbf{q}\mathbf{q_b}$ levels. Velocity-level constraints of sensitivity equations are fulfilled to a high extent. The constraint violation errors at the position-level exhibit similar trend as in the case of the adjoint method presented in fig 5.2.

There are a few comments that should be highlighted here to address some practical issues associated with the implementation of the adjoint method. It was observed that a grid density at which the state variables are recorded in memory during forward integration significantly affects the accuracy of the adjoint method. In this example, a uniform

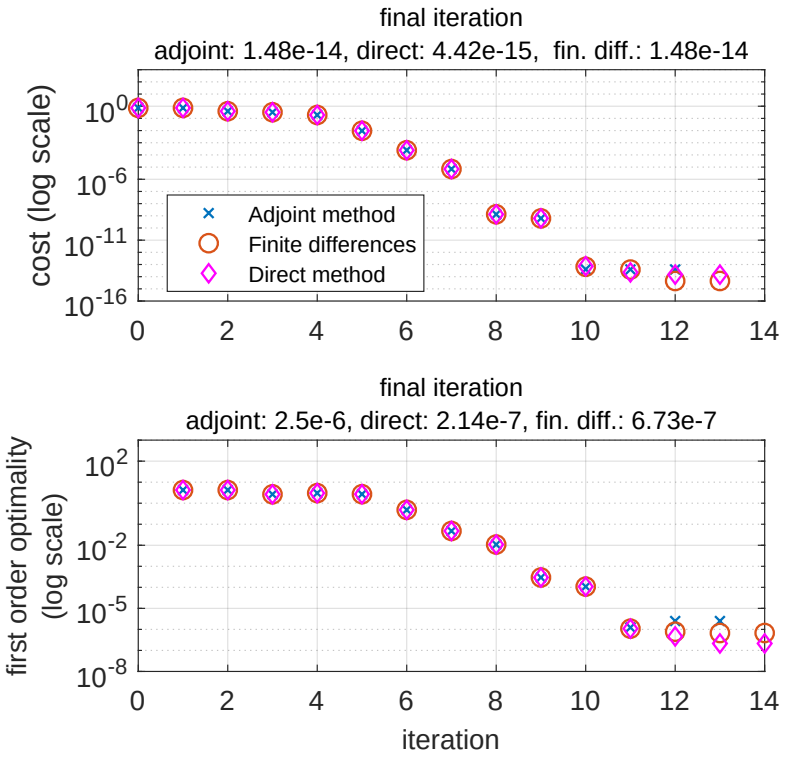


Figure 5.5: Cost function and first-order optimality measure for different approaches

grid of 1001 points has been used, which implies that the forward dynamics solutions are stored at each time step $\Delta t = 0.005$ s. Crucially, a visible loss of accuracy of the adjoint method was experimentally registered for $\Delta t > 0.02$. Usually the time grid used in the forward integration of the EOM is not consistent with the grid exploited by the backward scheme. This necessitates the usage of some kind of interpolation strategy to generate the state variables at particular time instant on demand whenever there is a request from backward integration scheme. Herein, we use a cubic spline interpolation, which proved to be sufficiently accurate.

Table 5.2 shows numerical values of the gradient of the performance measure calculated by four approaches: the Hamiltonian based adjoint and direct differentiation methods, finite differences, and complex-step methods. One can see a high agreement between the results. Particularly, the values obtained by direct differentiation and complex-step methods agree up to 7 significant digits. On the other hand, the gradient obtained by the Hamiltonian adjoint method retains high precision while reducing the time of calculations for the test case presented here. This property can be observed by comparing the time of execution of the examined approaches. The table presents the accumulated cost of solving forward and adjoint systems, whereas the direct method involves the cost of simultaneous solution of state and sensitivity equations. An additional numerical quantity expressed

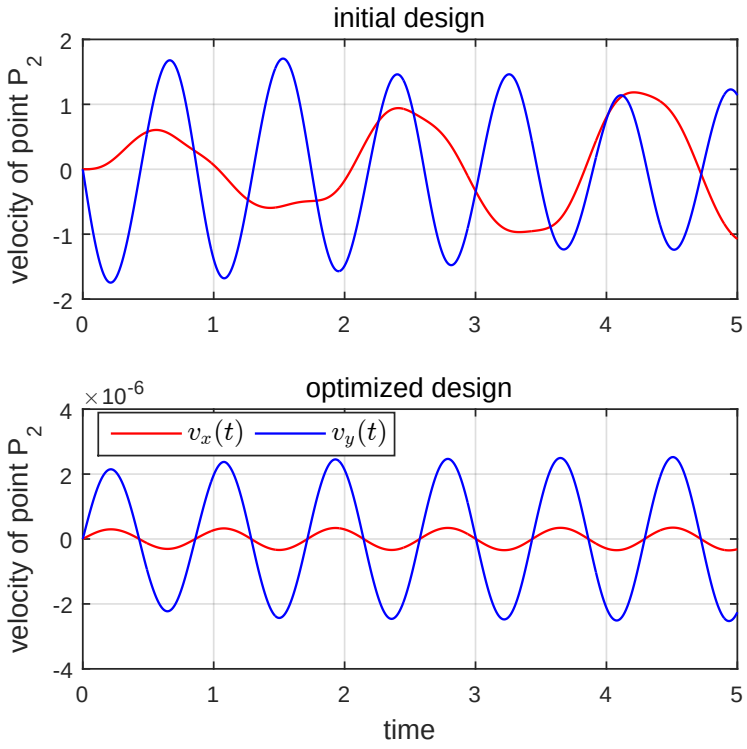


Figure 5.6: Velocity of point $\mathbf{P}_2$ for non-optimized and optimized parameters[1]

in percentages and shown in the third column of the table refers to the sole execution of the adjoint method, excluding the execution time of the forward dynamics problem. The last two columns present the absolute and relative errors made in the gradients with respect to the complex-step approach. The method has been picked as a reference due to its adequate precision and implementational simplicity. The absolute errors for the adjoint method deviate most heavily as compared to the other methods. However, it should be pointed out that even the complex method is prone to numerical errors and the exact values are not known precisely in this case. Moreover, it is indicated that the adjoint method outperforms its counterparts, particularly, when the solver tolerances are lowered. The results in tab. 5.2 agree with the data available in the literature [36], where the gradient was numerically calculated and approximated as $\nabla J = [-4.2294, 3.2092]^T$ for the same five-bar, which is used here. Moreover, a classical acceleration-based formulation has been employed [77, 90], which yielded the adjoint DAE that was solved by means of the constant step trapezoidal rule. This approach returned the following result $\nabla J = [-4.2359607, 3.2150296]^T$ and took 4.61 seconds to finish the calculations.

The interior point method is employed as an optimization procedure. An optimal design problem for the fivebar mechanism is solved by the adjoint-based, direct, and

[1]Unless stated otherwise, all plotted quantities are expressed in SI units.

complex-step differentiation methods. The comparative results are depicted in fig. 5.5, which presents the history of the cost function and first-order optimality measure [97] versus the number of iterations performed by the optimizer. The optimal solution found by the numerical optimization procedure aided by the Hamiltonian based adjoint method is $\mathbf{b}^{\mathbf{opt}} = [2.4319236, 1.8448689]^T$ m. Additionally, one can come up with the reference solution by forming a static equilibrium conditions. This yields a linear system of equations with prescribed position vector, in which reaction forces and free lengths are treated as unknowns. Both solutions agree with each other up to 8 significant digits.

The comparison between initial and optimized response of the MBS is presented in fig. 5.6. Red and blue lines correspond to the linear velocities of point $\mathbf{P}_2$ in the x and y direction, respectively. Please note that due to the numerical errors, especially constraint violation errors and lack of damping in the system, there may appear some small discrepancy in the desired velocity that ought to be identically zero. According to fig. 5.5, the optimization procedure has taken 14 or fewer iterations to find a solution in order to produce the results depicted in fig. 5.6 (dependent on the choice of the approach).

Table 5.2: Comparative results for various approaches – position-based optimization

Method Tol. $= 10^{-5}$	$\frac{\partial J}{\partial b_1}$	$\frac{\partial J}{\partial b_2}$	time (s) (adjoint %)	abs. error	rel. error
Hamiltonian adjoint method	-4.2280067	3.2102194	3.43 (50.22%)	$1.79 \cdot 10^{-4}$	$3.37 \cdot 10^{-5}$
Hamiltonian direct diff. method	-4.2280994	3.2103914	8.68	$1.84 \cdot 10^{-5}$	$3.46 \cdot 10^{-6}$
complex-step	-4.2280978	3.2103731	7.68	×	×
finite differences	-4.2280859	3.2104095	11.14	$3.83 \cdot 10^{-5}$	$7.21 \cdot 10^{-6}$
Tol. $= 10^{-7}$					
Hamiltonian adjoint method	-4.2280794	3.2102713	7.05 (61.72%)	$1.65 \cdot 10^{-4}$	$3.11 \cdot 10^{-6}$
Hamiltonian direct diff. method	-4.2281233	3.2104310	8.95	$3.16 \cdot 10^{-7}$	$5.96 \cdot 10^{-8}$
complex-step	-4.2281232	3.2104307	10.72	×	×
finite differences	-4.2281226	3.2104316	15.21	$1.08 \cdot 10^{-6}$	$2.04 \cdot 10^{-7}$

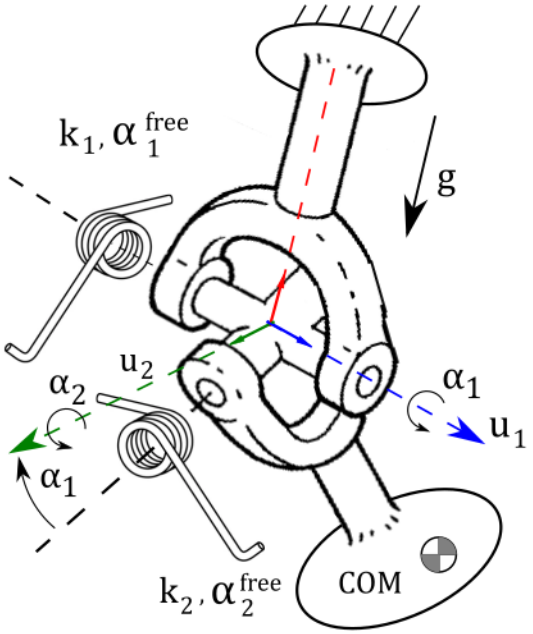


Figure 5.7: The spatial pendulum with torsional springs attached to ground via a Hooke joint

5.3 Optimal design of a spatial pendulum

In this example a pendulum moves in the gravitational field with two torsional springs attached to its fixture (c.f. fig. 5.7). The goal is to find static equilibrium conditions by selecting suitable free lengths $\alpha_1^{free}, \alpha_2^{free}$ of the springs, which are treated as design variables $\mathbf{b}$. This can be stated as an optimal design problem with the following cost functional:

$$J = \frac{1}{2}\int_0^{t_f} (\mathbf{r} - \mathbf{r}^{\text{init}})^T (\mathbf{r} - \mathbf{r}^{\text{init}})\, \mathrm{d}t, \tag{5.2}$$

where $\mathbf{r}$ locates the center of mass in the global frame, and $\mathbf{r}^{(0)}$ refers to the initial configuration of the body. Although the MBS consists of only a single body, the dynamic and kinematic relations arising from the springs and the universal joint are far from trivial. For example, the torque from both springs acting on the body can be calculated as:

$$\boldsymbol{\tau}^{\text{spring}} = k_1 \mathbf{u}_1 (\alpha_1 - \alpha_1^{free}) + k_2 \mathbf{A}^{\text{int}} \mathbf{u}_2 (\alpha_2 - \alpha_2^{free}), \tag{5.3}$$

where $k_1 = k_2 = 20$ Nm are the springs' stiffness, $\mathbf{u}_1 = [1,0,0]^T, \mathbf{u}_2 = [0,0,1]^T$ denote the springs' axes expressed in the global (parent body) reference frame, and the term $\mathbf{A}^{\text{int}}$ indicates the rotation about $\mathbf{u}_1$ axis by α_1. Joint rotations $\boldsymbol{\alpha} = [\alpha_1, \alpha_2]^T$ must be performed in the correct order, and the matrix $\mathbf{A}^{\text{int}}$ ensures that the second term of the torque is calculated about appropriate axis, denoted by green line in fig. 5.7.

The optimization (using interior-point algorithm) started with initial guess $\mathbf{b}_0 = [\frac{\pi}{5}, \frac{\pi}{7}]^T \approx [0.62832, 0.4488]^T$ converged to a solution in 14 iterations as indicated in fig. 5.8 and yielded the following result: $\mathbf{b}^{\text{opt}} = [1.0106, 0.65362]^T$. Figure 5.9 compares joint velocities of the pendulum recorded for the initial guess $\mathbf{b}_0$ and the optimized system $\mathbf{b}^{\text{opt}}$.

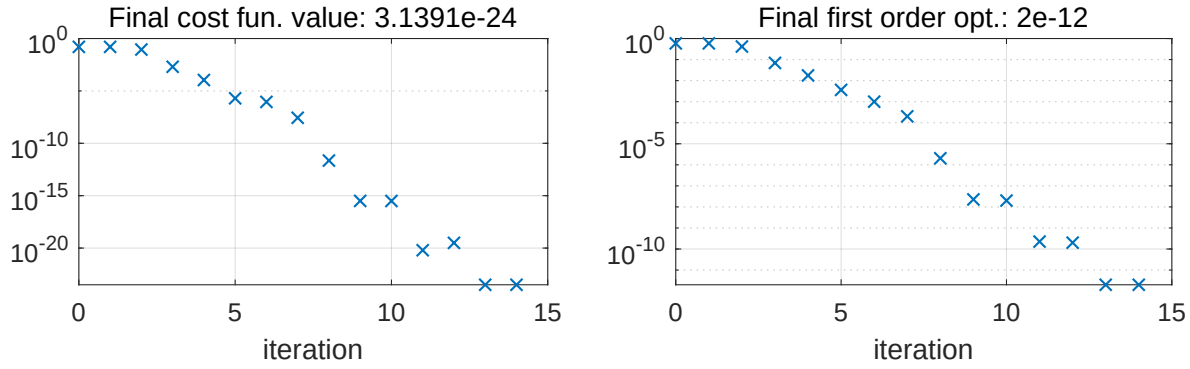


Figure 5.8: Convergence results: cost and first-order optimality measures

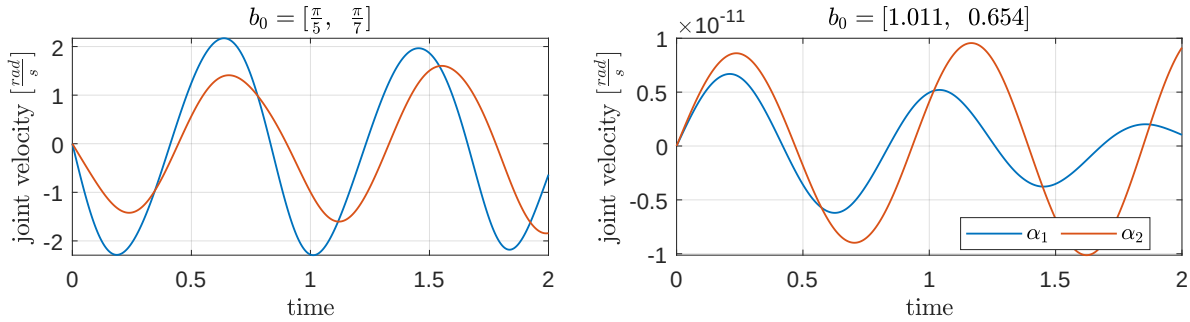


Figure 5.9: Joint velocities of the pendulum for non-optimized and optimized system (please mind the 10^{-11} factor on the right)

The adjoint variables required to evaluate the gradient $\nabla_{\mathbf{b}} J_{\mathbf{b}}$ may be calculated in two distinct ways: either by solving eq. (4.14) directly or by integrating joint-space system (4.29). In the latter case, absolute adjoint coordinates $\boldsymbol{\eta}, \boldsymbol{\xi}, \boldsymbol{\mu}$ can be obtained from eqs. (4.26). Figure 5.10 displays the values of the adjoint variables at the initial step of the procedure (i.e., for the design variables equal to $\mathbf{b}_0$) calculated via both approaches. The comparison clearly shows that the adjoint costate variables remain consistent with their absolute-coordinate counterparts derived in section 4.3.

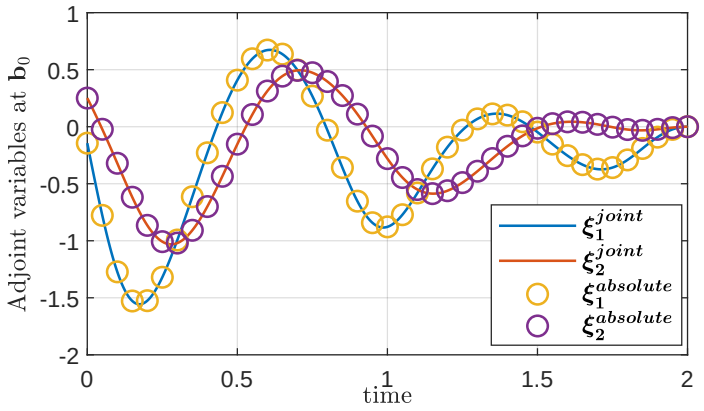


Figure 5.10: The adjoint variables recreated from the joint costate variables compared with their absolute-coordinate counterparts (4.14) (dots)

5.4 Optimal control of a double pendulum on a cart

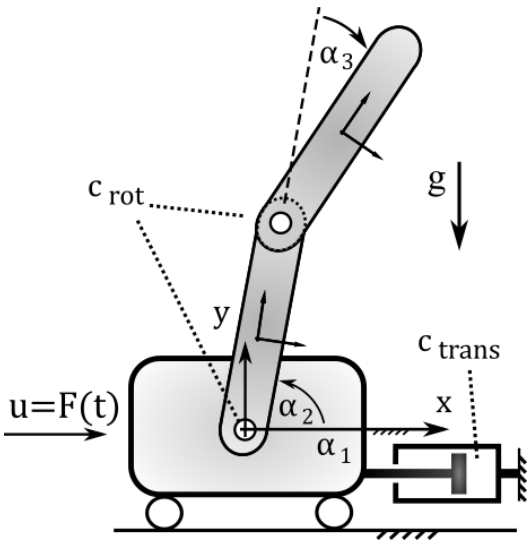


Figure 5.11: Double pendulum on a cart

Table 5.3: Input data

Symbol	Value / units
m_{cart}	0.8 kg
J_z^{cart}	0.019 kg m^2
l_{pend}	1 m
m_{pend}	0.5 kg
J_z^{pend}	0.043 kg m^2
c_{trans}	0.5 $\frac{\text{kg}}{\text{s}}$
c_{rot}	0.3 $\frac{\text{kg m}^2}{\text{s}}$
$u_0(t)$	$\equiv 0$ N
t_f	3 s

In this subsection a planar mutibody system that consists of a double pendulum attached to a cart is investigated. The gravity acts along the negative y axis of the global frame, and a viscous friction loads are introduced at each joint in the system. The mechanism is controlled by a horizontal force applied to the cart. The vector of design parameters $\mathbf{b}$ consists of discretized input signals $u(t)$, where $\Delta t = 0.005$ s is the discretization step. Consequently, $\mathbf{b} = [b_0, b_1, \cdots, b_N]$, where $b_0 = u(0)$, $b_1 = u(\Delta t), \cdots, b_N = u(N\Delta t) = u(t_f)$ and $N = \frac{t_f}{\Delta t}$. The input signal $u(t)$ can be easily recreated from $\mathbf{b}$. When the integration procedure requests an intermediate value, a spline interpolation is performed to approximate it. The parameters for the system, such as masses, moments of inertia, lengths, etc., are gathered in table 5.3. The quantity $u_0(t)$ shown in the table represents

the initially-assumed input force to the model. Although the analyzed multibody system moves in plane, it has been modeled with a spatial approach following the methodology presented in sections 3.2.2 and 3.3.1. Two different scenarios will be considered: a swing-up maneuver and a stabilization of the cart with pendula freely falling in the gravitational field.

5.4.1 Swing-up maneuver

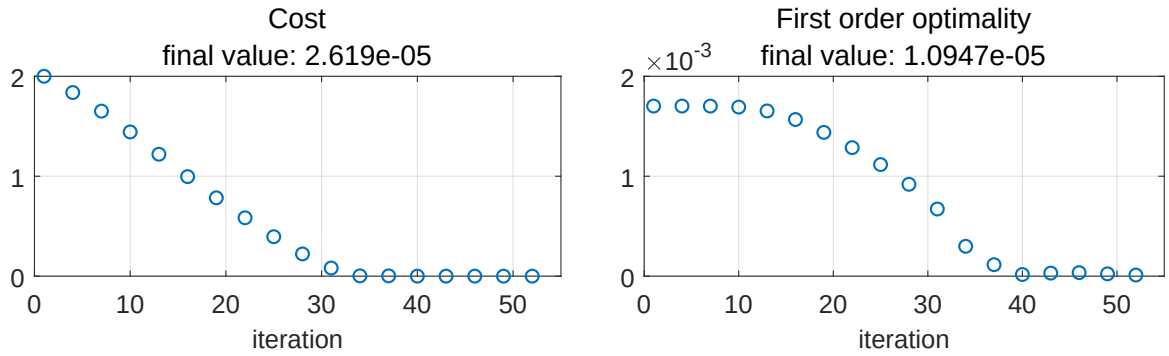


Figure 5.12: Cost function and first-order optimality measure of the swing-up maneuver optimization when $\gamma = 0$

Initial conditions for joint angles are specified to be $\alpha_1 = 0, \alpha_2 = -\frac{\pi}{2}, \alpha_3 = 0$, and translate to the system stable equilibrium. The goal of this task is to stabilize the open-loop chain in the upright configuration. Mathematically, this can be described in the following way:

$$J = \frac{1}{2}\left(\alpha_2 - \frac{\pi}{2}\right)^2 + \frac{1}{2}\alpha_3^2 + \frac{\gamma}{2}\left(\dot{\alpha}_2^2 + \dot{\alpha}_3^2\right) \quad \text{for } t = t_f, \tag{5.4}$$

where γ is a boolean value enforcing that pendula will stop moving at the final time if $\gamma = 1$. The initial guess is simply no actuation ($u_0(t) = 0$) exerted on the cart. The gradient is calculated by solving eq. (4.29) and conveyed to the optimization algorithm. The employed procedure utilizes `ode45` Matlab function to integrate forward dynamics and the adjoint system, and `fminunc` to compute the desired control input. Figure 5.12 presents the optimization convergence rates, specifically, how the cost and first-order optimality measure progressed for the iteration count when $\gamma = 0$. The first 30 iterations were performed with the steepest descent method, which uniformly reduced the cost to a near-zero value. Subsequently, we employed the quasi-Newton approach with the BFGS method for updating the Hessian for better convergence. In the following step, the optimization has been re-run with the latest generated input function used as the initial guess for the case $\gamma = 1$ in eq. (5.4). The progress of this computation is presented in

fig. 5.13, while the results of both processes are displayed in figure 5.14. The plotted lines represent the absolute angle ($\varphi_2 = \alpha_2, \varphi_3 = \alpha_2 + \alpha_3$) of both pendula at the final iteration for two cases: $\gamma = \{0, 1\}$. Let us point out that the solid lines enter the final value with a slope, which is zero, suggesting that the final velocity equals zero.

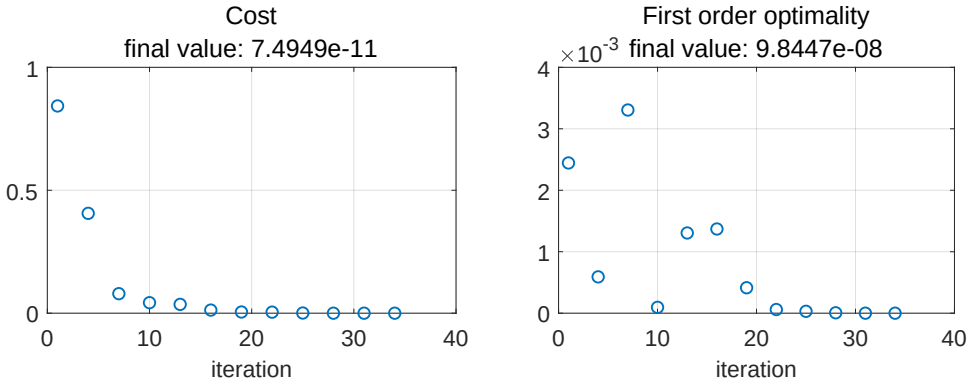


Figure 5.13: Cost function and first-order optimality measure of the swing-up maneuver optimization when $\gamma = 1$

Similarly to what has been presented in sec. 5.3, the gradient of the objective function computed at the first iteration from ODE (4.29) is compared with its DAE counterpart (c.f. eq. (4.11)). The absolute adjoint variables can be recreated from the joint-space by using a set of equations (4.26). Figure 5.15 shows that these quantities have almost identical values. Moreover, we append a plot with the results taken from complex-step method, which additionally shows a good agreement. Lastly, figure 5.16 displays constraint violation errors at different levels: velocity-level $\boldsymbol{\Psi}$, its first derivative $\dot{\boldsymbol{\Psi}}$, as well as $\ddot{\boldsymbol{\Psi}}$ (c.f. tab. 4.3). The plots refer to the ODE (4.29) and DAE (eq. (4.11)) formulations, respectively. The latter plot reveals that the constraint equations $\ddot{\boldsymbol{\Psi}} = \mathbf{0}$ is fulfilled up to the machine accuracy since it is enforced explicitly in eq. (4.11b). On the other hand, lower-level constraints are characterized by a much larger constraint violation errors, when compared against the ODE formulation.

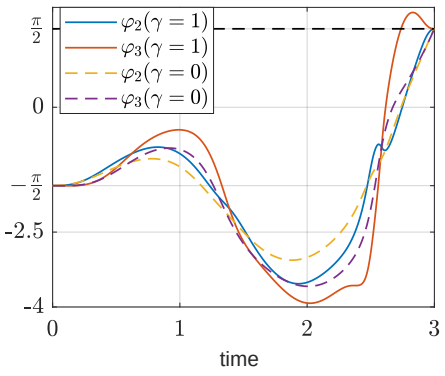


Figure 5.14: Absolute orientation of the pundulums attached to the cart

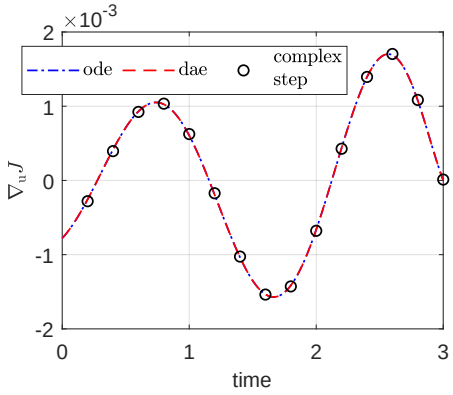


Figure 5.15: The results of different gradient evaluation methods at the initial step ($u(t) = 0$)

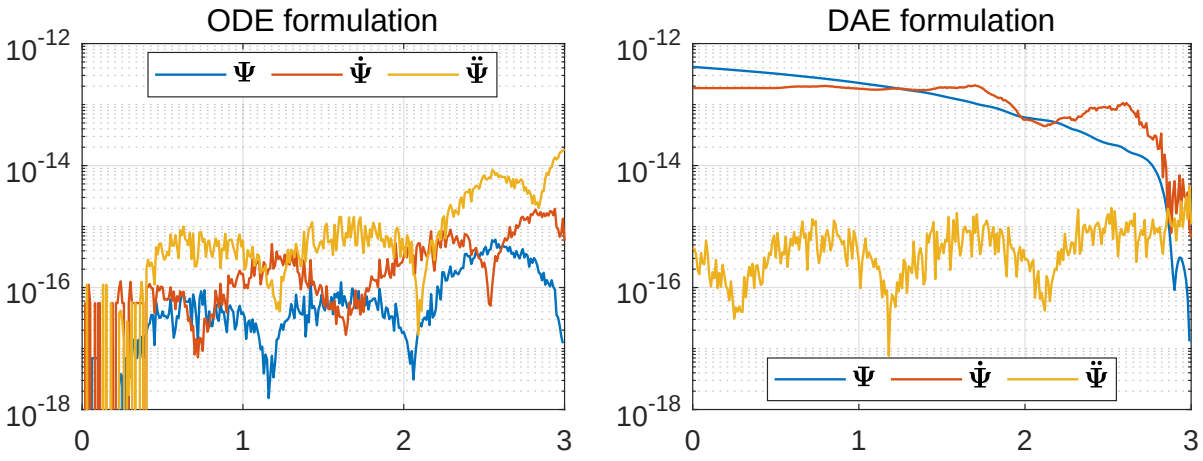


Figure 5.16: Constraint violation errors for adjoint-based conditions at $\mathbf{u}_0$. Comparison between joint-space (4.29) and absolute (4.11) formulations

5.4.2 Stabilization of the cart

This test case investigates the calculation of the input control signal that will stabilize the cart while the pendula are free-falling under the gravity forces. The initial configuration of the MBS can be described in the following way: $\alpha_1 = 0$, $\alpha_2 = \frac{\pi}{4}$, $\alpha_3 = 0$, while the bodies' velocities are equal to zero. Contrary to the previous example, this problem can be posed in two distinct ways. The performance measure is defined in the following way:

$$J = \frac{1}{2}\int_0^{t_f} x^2 \, \mathrm{d}t + \frac{1}{2}x^2|_{t_f}, \tag{5.5}$$

where $x := \alpha_0$ denotes the x-coordinate of the cart. On the other hand, it is possible to add a driving constraint equation of the form $\Phi^D \equiv x = 0$, solve only the forward dynamics problem, and record the Lagrange multiplier $\lambda_{x=0}$ associated with the appropriate reaction force. The specified approach yields a very good approximation of the solution and can be utilized to compare and verify the outcome. Figure 5.17 depicts the input control forces calculated by using the approach proposed in this thesis and by looking at the driving constraint force. Consequently, figure 5.18 presents how the position of the cart changes in time for different actuation signals. The response of the cart for the optimized input signal diverges ± 7 mm from the equilibrium, which is a very good fit.

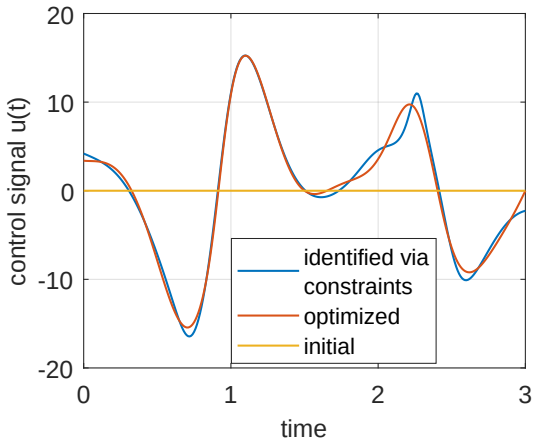


Figure 5.17: Control signal for different test cases

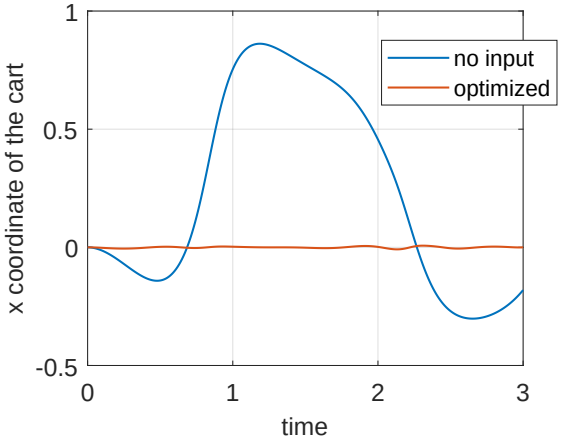


Figure 5.18: Motion of the cart for different control signals

The starting guess for the optimization is again set to be zero, i.e., $u_0(t) = 0$. Figure 5.19 shows the convergence rate of the `fminunc` Matlab function, which executes quasi-Newton algorithm with a user-supplied gradient calculated from equation (4.29). Figure 5.20 displays the gradients computed at the initial guess of the optimization (left plot) and evaluated for the solution found from utilizing a driving constraint (right plot). One can see a high agreement between the employed methods. Moreover, the value of the gradient evaluated at the solution is expected to be equal to zero. This is not precisely the case in figure 5.20; however, the magnitude of the gradient computed at the exact solution is four orders of magnitude lower when compared with the gradient evaluated at the initial guess.

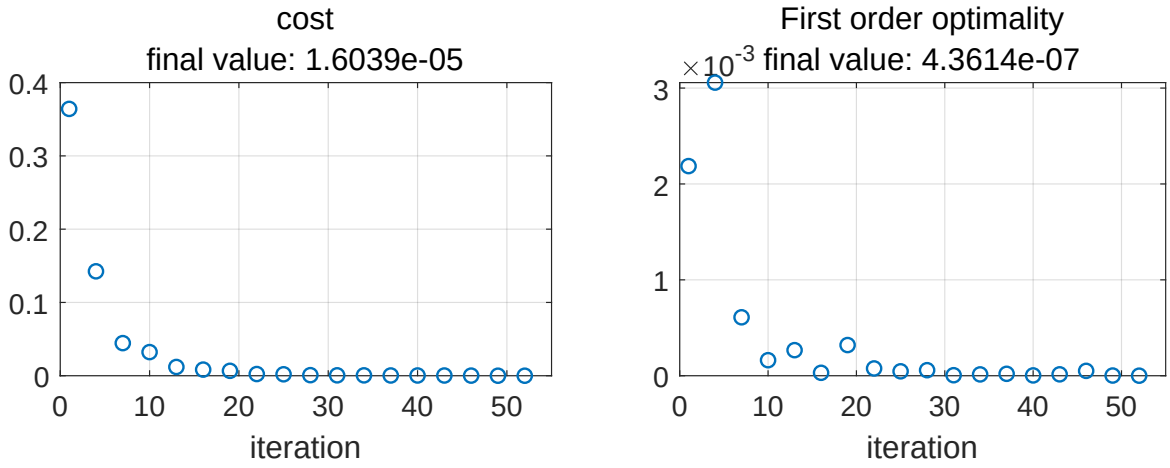


Figure 5.19: Cost function and first-order optimality measure for stabilization of the cart in rest

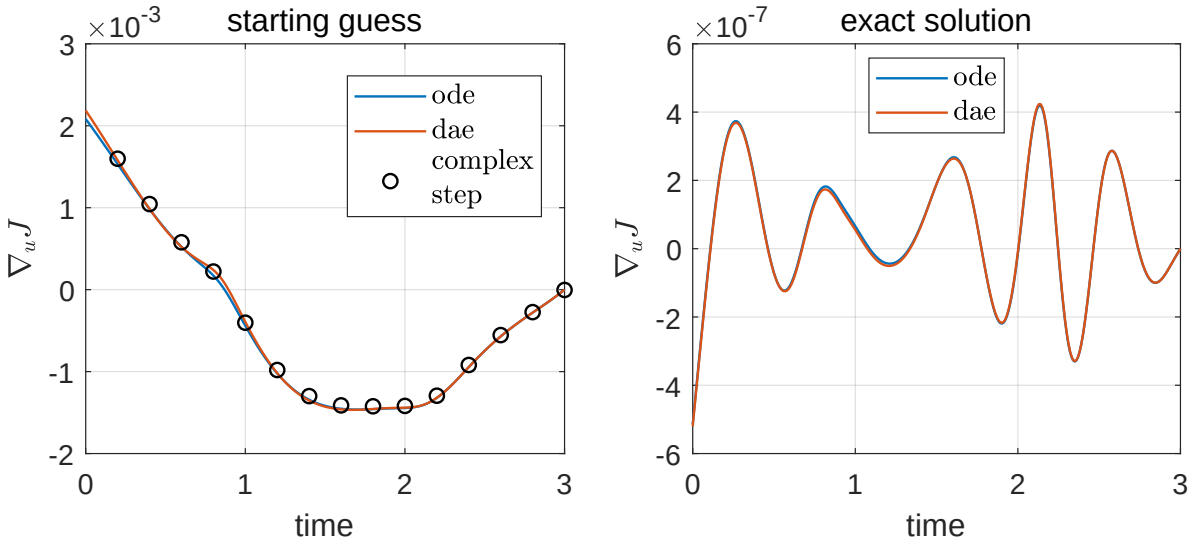


Figure 5.20: The results of different gradient evaluation methods at the first step and for the exact solution

5.5 Optimal control of the two-degree-of-freedom pendulum

The following section is devoted to finding appropriate input control signals for the spatial multibody system composed of the inverted pendulum and a five-bar linkage. The layout of the MBS is depicted in figure 5.21. The linkage is actuated by two DC motors that have been modeled by introducing additional state variables: motor shaft orientations $\varphi_{m_1}, \varphi_{m_4}$ and velocities $\dot{\varphi}_{m_1}, \dot{\varphi}_{m_4}$ conjugated with momenta $\mathrm{p}_{m_1}, \mathrm{p}_{m_4}$, respectively. Additionally, the transmission between the motors and the propelled bodies is modeled via constraint equation $\Phi^{\text{trans}} \equiv \varphi_{m_i} - k_g \cdot \varphi_i = 0$, $i = \{1, 4\}$. The torque produced by each of the motors is calculated with the following formula:

$$\tau_{m_i}(t) = \frac{k_t \cdot (V_i(t) - k_m \cdot \dot{\varphi}_{m_i}(t))}{R_m} - B_m \cdot \dot{\varphi}_{m_i}(t), \tag{5.6}$$

where the constant parameters are explained in table 5.4, and $V(t)$ is a time-varying function of the applied voltage. Model (5.6) is simplified by omitting the term associated with motor inductance since it is relatively low compared to the resistance. When solving the dynamics equations, we set either $u_1(t) = V_1(t)$ or $u_2(t) = V_4(t)$, where the discretized input signal can be represented as $\mathbf{u}_i = [b_{i0}, b_{i1} \cdots, b_{iN}]^T$. Any two consecutive entries of $\mathbf{u}_i$ are Δt apart on the time axis. A cubic spline interpolation is employed when the integrator requests an intermediate input signal value. Hence, the number of design variables is equal to $\dim(\mathbf{b}) = k = 2 \cdot (\frac{t_f}{\Delta t} + 1)$. The dynamics of DC motors is introduced

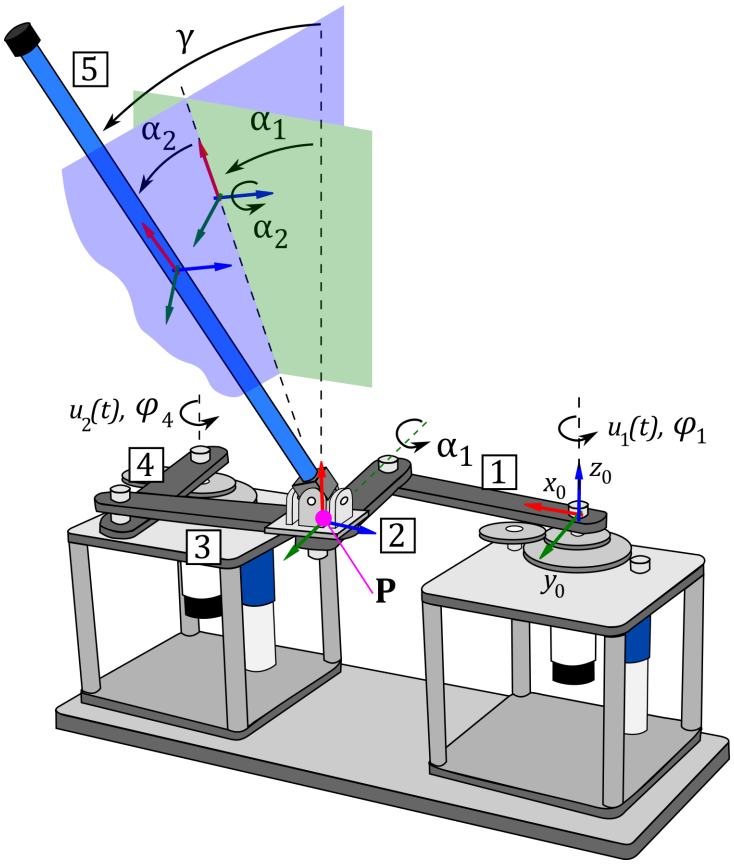


Figure 5.21: Spatial multibody system with inverted 2-DOF pendulum on a five-bar linkage

by appending EOM (3.14) with the following equations (for $i = \{1, 4\}$):

$$\mathrm{p}_{m_i} = J_m \dot{\varphi}_{m_i} + (\Phi^{\mathrm{trans}}_{\mathbf{q}})^T \sigma_{m_i}, \qquad \dot{\mathrm{p}}_{m_i} = \tau_{m_i} \tag{5.7}$$

A physical pendulum is attached to the five-bar linkage at point **P** via a Hooke joint. The configuration of the pendulum can be conveniently described by means of joint coordinate $\boldsymbol{\alpha}_5 = [\alpha_1, \alpha_2]^T$, which has been demonstrated in fig. 5.21. Symbol $\gamma = \mathrm{acos}\left(\bar{\mathbf{z}}^T \mathbf{A}_5 \bar{\mathbf{x}}\right)$ denotes absolute value of the angle between pendulum's (local x) and vertical (global z) axes. Body 5 (pendulum) orientation matrix with respect to the global coordinate frame is denoted by $\mathbf{A}_5$, whereas $\bar{\mathbf{x}}, \bar{\mathbf{z}}$ are vector representations of the x, z axes in the global coordinate frame.

The dynamic properties of all bodies have been gathered in table 5.4. In the following subsections we will investigate two distinct scenarios: a vertical stabilization of the pendulum in a fixed time interval and a stabilization of the linkage influenced by the inertial forces from the falling pendulum in the gravitational field.

Table 5.4: Model parameters

Parameter	Value
Length of bodies 1–4	$l_i = 0.127$ m (5 inch)
Mass of bodies 1–4	$m_i = 0.065$ kg
Moment of inertia of bodies 1–4	$J_z = 9 \cdot 10^{-5}\,\mathrm{kg\,m^2}$
Length of pendulum	$l_5 = 0.3365$ m
Mass of pendulum	$m_5 = 0.125$ kg
Moment of inertia of pendulum	$J_x = 6.5 \cdot 10^{-6}\,\mathrm{kg\,m^2}$ $J_y = J_z = 1.8 \cdot 10^{-4}\,\mathrm{kg\,m^2}$
Initial position of $\mathbf{P}$	$\mathbf{P}^{(0)} = (0.127, 0.127, 0)$ m
Discretization step	$\Delta t = 0.005$ s
Motor model parameters	
Moment of inertia of the motor shaft	$J_m = 4.6 \cdot 10^{-7}\,\mathrm{kg\,m^2}$
Motor back-emf constant	$k_m = 7.68 \cdot 10^{-3}\,\frac{\mathrm{V\,s}}{\mathrm{rad}}$
Motor current - torque constant	$k_t = 5.3 \cdot 10^{-3}\,\frac{\mathrm{Nm}}{\mathrm{A}}$
Motor resistance	$R_m = 2.6\,\Omega$
Equivalent friction coef.	$B_m = 3.42 \cdot 10^{-6}\,\frac{\mathrm{Nm\,s}}{\mathrm{rad}}$
Transmission gear ratio	$k_g = 70$

5.5.1 Vertical Stabilization of the spatial 2 DOF pendulum

The initial position of all bodies is fixed, where the consecutive links of the five-bar are perpendicular to each other, and the initial orientation of the pendulum is slanted $\boldsymbol{\gamma}$ degrees from the vertical axis (by prescribing appropriate values for $\boldsymbol{\alpha}_5$). The objective is to actuate the five-bar linkage in such a way that the final configuration of the pendulum remains in the upright position and its velocity is minimal. Simultaneously, we want to avoid bifurcations of the five-bar linkage configurations, which occur, e.g., under hyper-extension of its two adjacent links. To this end, the following performance measure is proposed:

$$J = \int_0^{t_f} \frac{1}{2}\Big(\mathbf{P} - \mathbf{P}^{\mathrm{init}}\Big)^T \Big(\mathbf{P} - \mathbf{P}^{\mathrm{init}}\Big) dt + \frac{1}{2}\gamma^2|_{t_f} + \frac{w}{2}\mathbf{v}_5^T\mathbf{v}_5|_{t_f}, \tag{5.8}$$

where $\mathbf{v}_5$ refers to the spatial velocity of the pendulum, and w is a weight associated with the end time velocity criterion. Moreover, $\mathbf{P}^{\mathrm{init}}$ describes the initial coordinates of point $\mathbf{P}$.

One of the main issues of solving inverse dynamics problems via optimization methods is finding the correct initial guess. In this example, we iterate over four cases that can be treated as problems on their own or subproblems (SPs) leading to the main goal, i.e., stabilizing the pendulum tilted $\boldsymbol{\gamma} = 18°$ from the upright position.

- subproblem 1: $\boldsymbol{\alpha}_5 = [-5°, -5°]^T \quad \rightarrow \boldsymbol{\gamma} \approx 7°, w = 0$

- subproblem 2: $\boldsymbol{\alpha}_5 = [-10°, -10°]^T \rightarrow \gamma \approx 14°, w = 0$
- subproblem 3: $\boldsymbol{\alpha}_5 = [-10°, -15°]^T \rightarrow \gamma \approx 18°, w = 0$
- subproblem 4: $\boldsymbol{\alpha}_5 = [-10°, -15°]^T \rightarrow \gamma \approx 18°, w = 0.05$ (the main goal)

The first subproblem is initialized with $u_1(t) = u_2(t) \equiv 0$, its solution becomes the initial guess for the following SP, etc. It is worth noting that subproblems 2–4 refer to the control of the dynamical system beyond the applicability of a linearized mathematical model around the (unstable) equilibrium of the MBS. The final problem involves a non-zero weight w associated with the minimization of the terminal velocity of the pendulum.

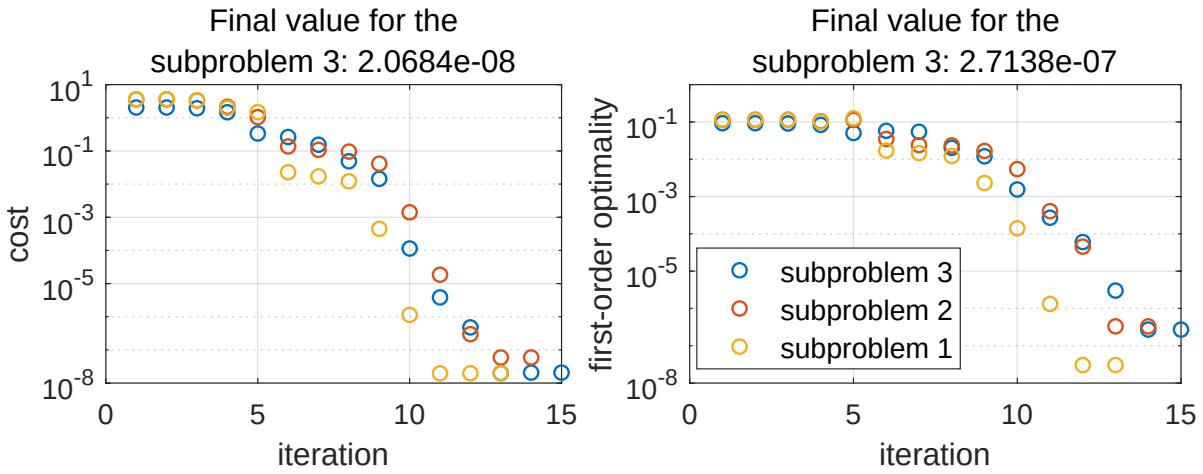


Figure 5.22: Cost function value and first-order optimality measure throughout the optimization for different initial configurations and with weight $w = 0$

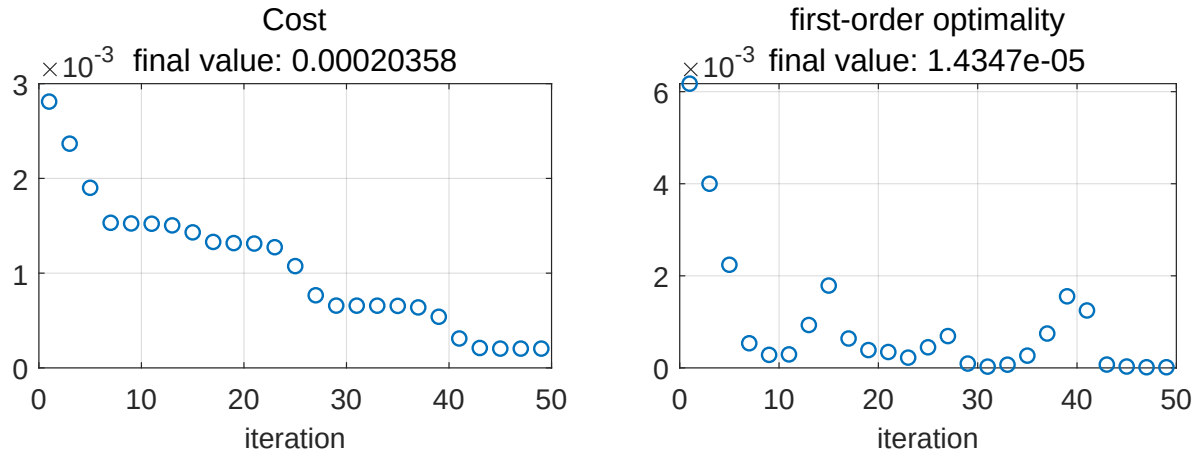


Figure 5.23: Cost function value and first-order optimality measure throughout the optimization for initial configuration $\boldsymbol{\alpha}_5 = [-10°, -15°]^T$ and $w = 0.05$ (SP4)

The total time of the maneuver is fixed to be $t_f = 0.5$ s, whereas the discretized data of the forward problem is stored in the computer memory with the constant time-step $\Delta t = 0.005$ s. This implies that the number of design parameters is equal to $\dim(\mathbf{b}) = k = 202$. Matlab sequential quadratic programming (SQP) procedure has been employed, which converged in 13–15 iterations in the cases of SPs 1–3 (see fig. 5.22). The final subproblem took significantly more iterations to complete, as implied by figure 5.23.

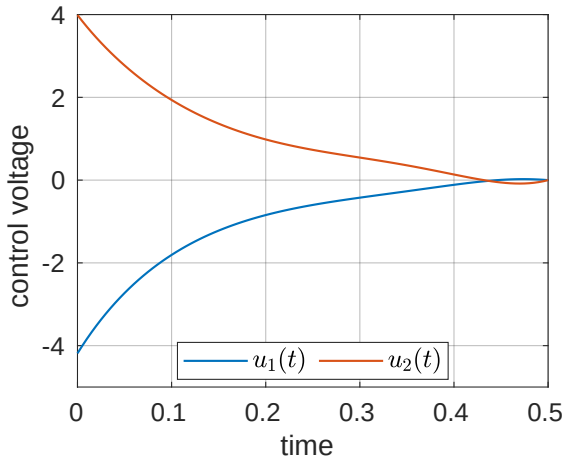


Figure 5.24: Computed input signals for the final subproblem that stabilize the pendulum in the vertical position

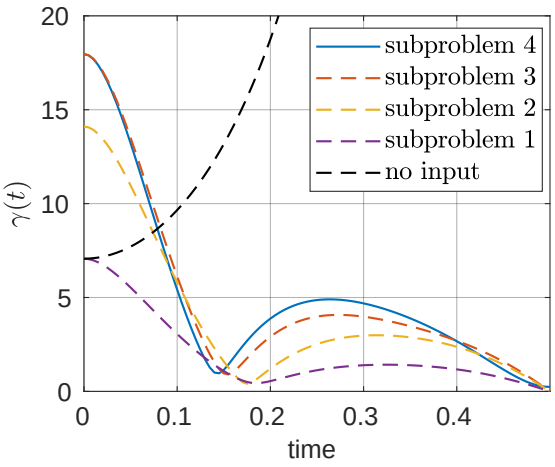
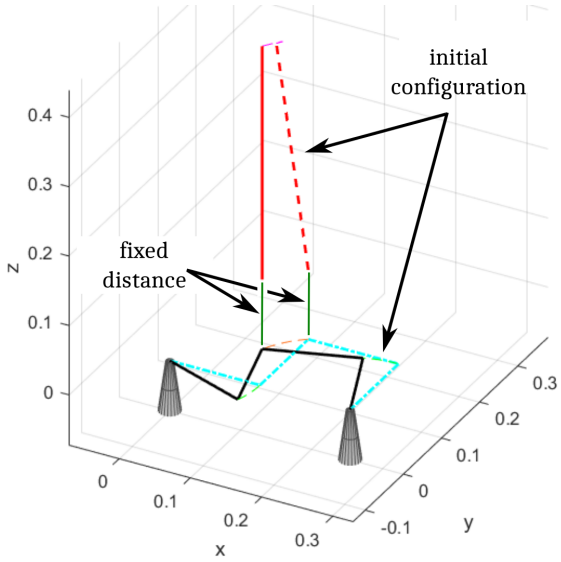


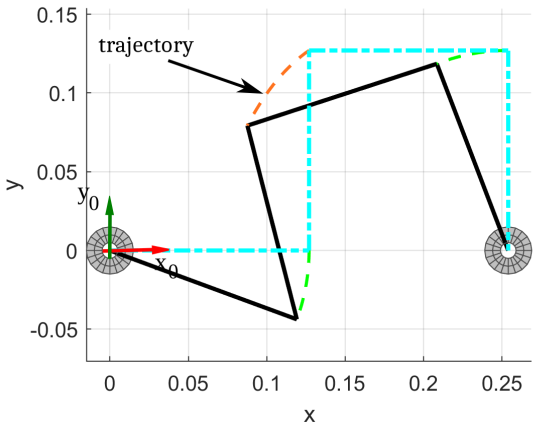
Figure 5.25: Angle between vertical and pendulum's axes for different input signals

Figure 5.24 shows the solution found by the numerical procedure for the SP 4, while the functions depicted in fig. 5.25 present the magnitude of γ obtained by solving each task. For comparison, the output from the simulation with no input was added (for the SP 1), which reveals that the unstabilized pendulum immediately falls under the gravity forces. We can also notice, that the ultimate scenario is finished with the value of γ approaching the final value in a tangential manner, which implies that the velocity is indeed close to zero. Furthermore, figures 5.26 and 5.27 depict the multibody system in the initial and terminal configurations for SPs 1 and 4, respectively.

Last but not least, figure 5.28 demonstrates the gradient computed at the first iteration for subproblem 2 (fig. 5.28a) and 3 (fig. 5.28b). The solid lines denote the result generated by the adjoint method. At the same time, the points on the plot represent the gradient approximated by the finite differences method with the perturbation step assigned to $\delta\, \mathrm{b}_i = 10^{-8}$. The close overlap between both results is a strong premise for a validity of the proposed method. On the other hand, specific values produced by the finite differences method rapidly diverge from the expected trend. This fact appears to be a numerical glitch (caused, e.g., by the inconsistency in initial or boundary conditions) since it affects only initial (or final) entries. Figure 5.28b includes additional samples in the initial section of



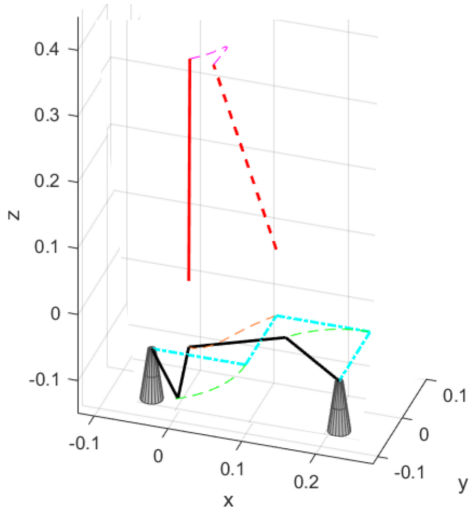
(a) Isometric view



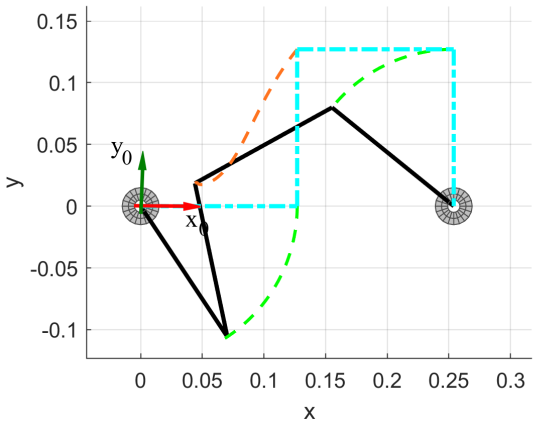
(b) Top-down view

Figure 5.26: Initial and final configuration of the MBS for SP1

the domain to verify this observation. Subsequently, fig. 5.28 reveals the efficiency gap between both methods, which manifests itself in the time of execution (not visible directly in the plots): the central differences method has computed a single point on the plot in a time exceeding 4 seconds (on a standard PC). On the other hand, the adjoint method computed the gradient for all 202 points in a similar time. The parameters of the testing platform are following: processor Intel Core i7 2700K (4 cores with 8 threads, clock rate: 3.5 GHz), memory with 8 GB RAM, operating system: Windows 7.

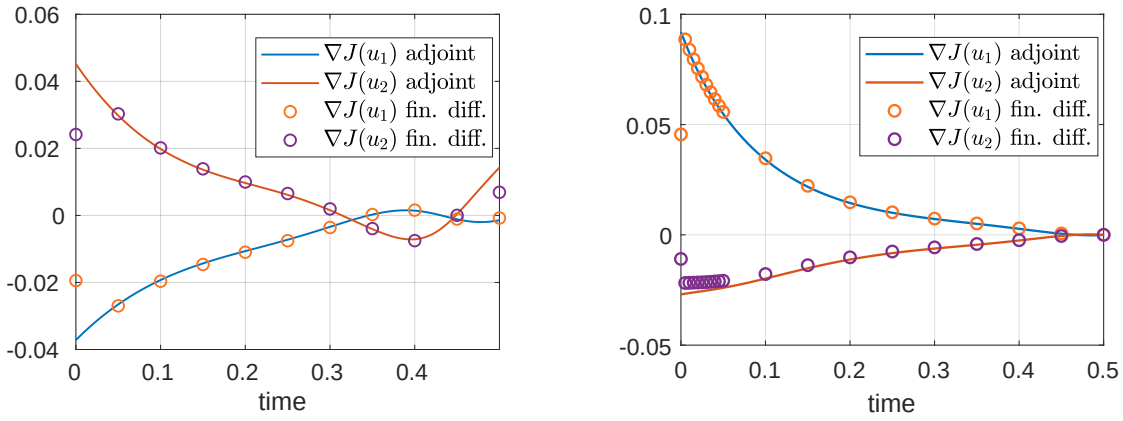


(a) Isometric view



(b) Top-down view

Figure 5.27: Initial and final configuration of the MBS for SP4



(a) SP2: $\mathbf{b}_0$ is equal to the solution of SP1 (b) SP3: $\mathbf{b}_0$ is equal to the solution of SP2

Figure 5.28: Gradient of the performance measure at the initial step of subproblems 2 (a) and 3 (b)

5.5.2 Minimization of the five-bar linkage oscillations

This scenario assumes that the pendulum's center of mass is below the plane of the five-bar linkage motion. Specifically, it is oriented as $\boldsymbol{\alpha}_5 = [160°, 10°]$ and free-falls under the gravity forces. As a result, the inertial forces exert motion on the five-bar linkage bodies. The goal is to compute motor voltages, that stabilize the linkage during the fixed time interval $t_f = 1.5$ s. The proposed performance measure that captures this behavior can be formulated in the following way:

$$J = \int_0^{t_f} \frac{1}{2}\Big(\mathbf{P} - \mathbf{P}^{\text{init}}\Big)^T \Big(\mathbf{P} - \mathbf{P}^{\text{init}}\Big). \tag{5.9}$$

Moreover, the following task can be solved in a similar fashion as presented in section 5.4.2, i.e., by prescribing additional driving constraints on the orientation of certain bodies; e.g., $\boldsymbol{\Phi}^D = [\varphi_1, \varphi_4]^T = \mathbf{0}$ (cf. fig. 5.17). The corresponding reaction forces accurately approximate the driving torques that guarantee proper trajectory. This approach can be used here to produce the reference results for comparison.

Figure 5.29 depicts the convergence history of the SQP algorithm, until the stopping criterion ($\frac{|\Delta \mathbf{b}_i|}{|\mathbf{b}_i|} < \varepsilon = 10^{-6}$) has been met. Subsequently, fig. 5.30 compares the response of the model (in the form of x, y coordinates of point $\mathbf{P}$) for two sets of signals: initial $\mathbf{b}_0 = \mathbf{0}$ and optimized $\mathbf{b}^{\text{opt}}$. One can see that without the external excitation, the damping introduced by the motors causes the MBS to stop in approximately 1 s. Conversely, when the voltage is applied to the motors, the point $\mathbf{P}$ remains nearly in place, as demonstrated in fig. 5.31a, where we see that pendulum's trajectory projected on the $x - y$ plane (depicted by the magenta line). Concurrently, figure 5.31b indicates that the linkage

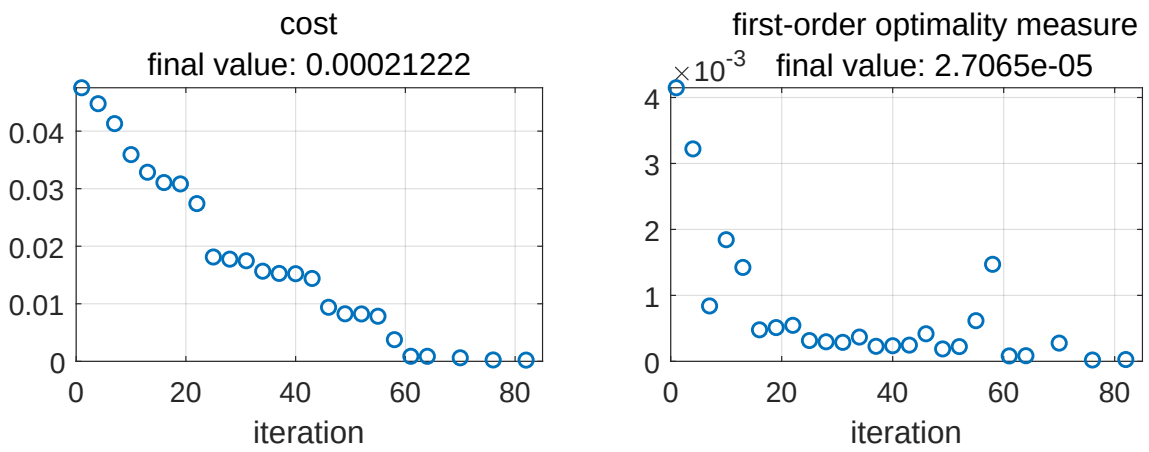


Figure 5.29: Cost function and first-order optimality measure throughout the SQP iterations

stops near its initial configuration; however, it moves away from its origin during the simulation.

Furthermore, the motor voltage obtained via optimization procedure supported with the adjoint method may be used to compute (via eq. (5.6)) driving torques acting on the terminal bodies of the linkage. Figure 5.32 presents these results compared with the outcome of the forward problem augmented with driving constraints. The shape of the optimized solution closely resembles the reference curves. In contrast, the discrepancies result from the iterative approach of the employed method. The optimization results can

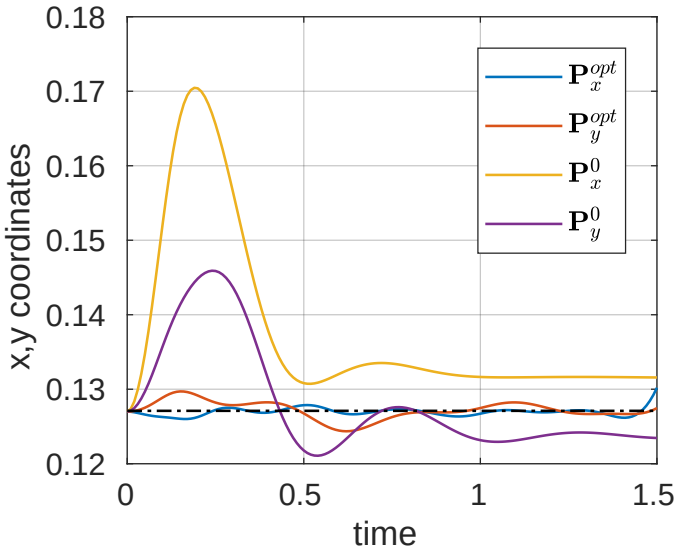
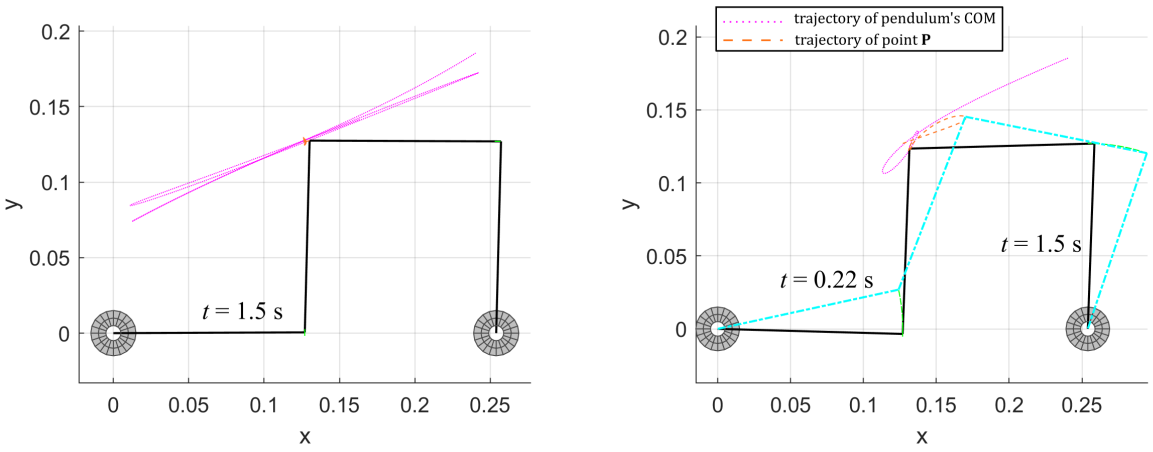


Figure 5.30: Coordinates of point **P** for initial and final input signals



(a) Final simulation (optimized input)

(b) Initial simulation: $u_1(t) = 0, u_2(t) = 0$

Figure 5.31: Top-down view on the trajectories of the MBS for initial and final simulations

become more accurate by tightening the tolerance of the solver at the cost of performing a larger amount of computations.

Ultimately, figure 5.33 depicts gradients computed at different iteration steps. The adjoint method has been again compared with the central finite differences method to verify the correctness of the implementation. One can observe a good consistency between the plots, although signal $\nabla J(u_1)$ is slightly offset in the case of the final iteration. Numerical inaccuracies may have caused this due to the fact that the gradient value near the solution is close to zero.

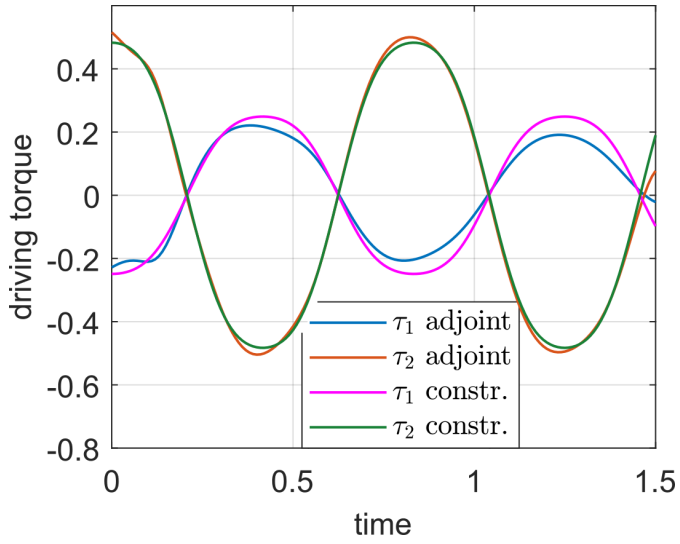


Figure 5.32: Applied actuation (torques) compared with accurate solution obtained via servo constraints $\mathbf{\Phi}^D = \mathbf{0}$

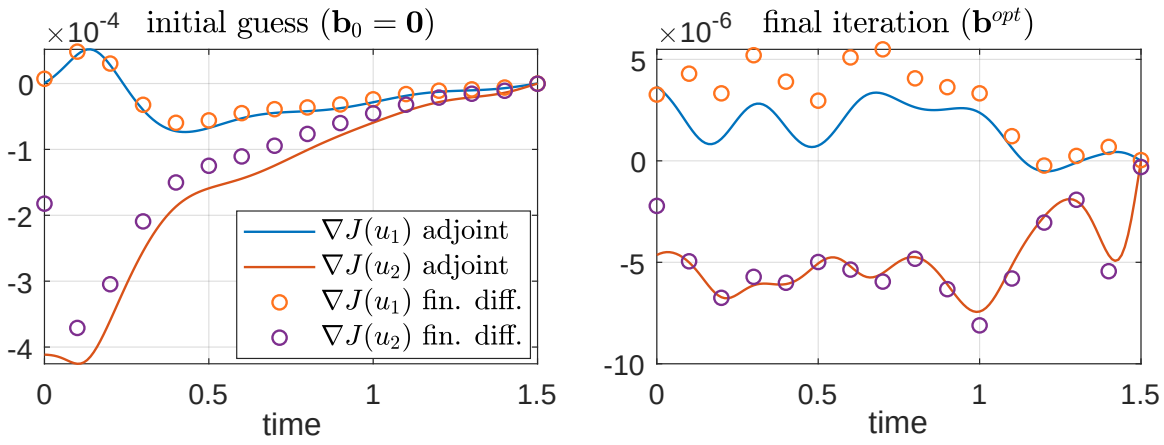


Figure 5.33: Gradient of the performance measure at the initial step and final iteration compared with finite differences method

5.6 Summary

The contents of this chapter have been divided into parts describing different problems that vary in difficulty and applicability in the real world. The first two examples presented simple benchmark mechanisms allowing for investigation of the properties and the validity of the proposed methods. Through numerical studies, it was possible to verify the validity of relations (4.26) as well as (4.22) developed in section 4.3. Applications show that the minimal-coordinate adjoint method is reliable, and the accuracy of computations is higher than the redundant-coordinate counterpart. The performance of the proposed approach has been quantified, especially in terms of constraint violation errors resulting from backward integration of the adjoint system.

The last example was based on a real object and was used to test the applicability of the developed methods in the spatial regime. The computed gradients proved accurate and reliable, whereas the numerical results have been rigorously validated by calculating the gradients using the complex-step approach and finite difference (or complex-step) method. The sensitivity information obtained from the proposed approaches is in good agreement with the mentioned methods, which prompts the Hamiltonian-based procedures derived herein to be good alternatives to existing methods when it comes to sensitivity analysis of multibody systems.

CHAPTER 6

PARALLEL IMPLEMENTATION OF THE ADJOINT METHOD

6.1 Introduction

Previous chapters (specifically sections 3.3 and 4.5) discussed mathematical formulations of complicated ODE and DAE systems (either dynamic or their adjoints) that can be solved in a parallel manner. This section will utilize the derived expressions in a practical implementation to test and present the results obtained from the proposed HADCA approach. A detailed algorithm explaining the specifics of the implementation will be outlined. Subsequently, the execution times of the specific parts of the proposed algorithm will be measured for the purpose of efficiency analysis. Optimal control of open-loop kinematic chain will serve as a basis of these investigations.

6.2 Problem description and implementation

Let us consider an open-loop planar kinematic chain consisting of a cart that performs horizontal motion and multiple pendula attached to the cart and each other. Figure 6.1 presents this setup and visualizes additional effects considered in the analysis, such as viscous friction in joints, gravity, and external force acting on the cart. This force actuates the whole system, and its discretized values constitute a set of optimization parameters. The main goal (similar to the one addressed in sec. 5.4.2) is to compute input force, which will stabilize the system in rest while pendula fall freely from their initial position under gravity forces. Consequently, the focus of this chapter is shifted to the single iteration of the optimization procedure, where the forward and adjoint systems are solved to compute the gradient of the performance measure. Throughout this chapter, we will assume the

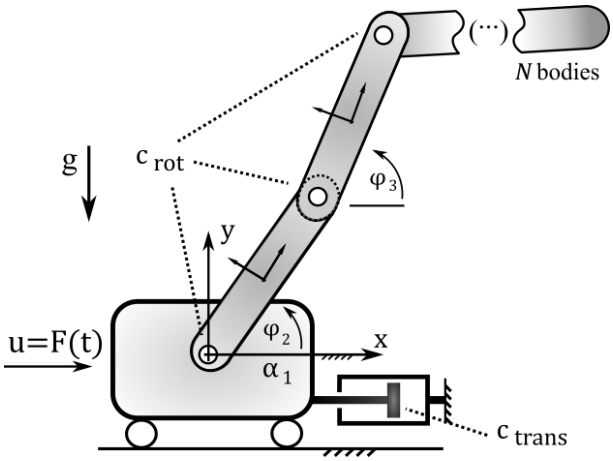


Figure 6.1: A multi-link pendulum on a cart as a benchmarked system

following initial configuration of the MBS: $x_1 = 0$ m, $\varphi_2 = \varphi_3 = \cdots = \varphi_{N_b-1} = \frac{\pi}{4}$ rad, whereas the initial velocity of all bodies is set to zero. The number of pendulum segments can be prescribed apriori, which is a convenient way of specifying the problem's size.

A research version (i.e., not including hardware- nor software-specific optimizations) of the computer code was written in C++ specifically for the purpose of benchmarking the HDCA/HADCA algorithms. The following libraries have been used as building blocks to develop the implementation:

- **eigen** library – matrix operations and linear algebra [52]
- **odeint** (boost library) – ODE numerical integration [1]
- **OpenMP** – compiler directives for data parallelism
- **nlopt** – optimization suite [116]
- `std::chrono` library – timing of the code execution

The code has been executed on a computing node equipped with two Intel Xeon Gold 6242 (clock 2.8 GHz, 2.2 Mb cache) processors and 192 GB RAM. Each processor has 16 physical cores with a multithreading capability, enabling the machine to 64 hyperthreads. The operating system is Ubuntu 20.04, whereas the software employed for compiling the code is *GNU Compiler Collection* (GCC) v. 9.4.

The core data structure of the developed implementation is the `Assembly` class. Each object of this class fulfills the role of an articulated body in the binary tree. Therefore, it involves the following attributes (objects derived from eigen library):

- $4 \times$ `Matrix3d` and $2 \times$ `Vector3d` – coefficients of the integral equation of motion (3.30)–(3.33)

- $2 \times$ `Vector3d` – bias terms of the acceleration analysis (4.50) (the matrices remain the same as in the point above due to the equivalence of the LHS in eqs. (3.14) and (3.15))
- $2 \times$ `Vector3d` – unknown reaction impulses in joints: $\mathbf{T}_1^A, \mathbf{T}_2^A$
- $2 \times$ `Vector3d` – unknown reaction forces in joints: $\mathbf{L}_1^A, \mathbf{L}_2^A$ (acceleration analysis)
- `Vector3d` – accumulated force vector $\mathbf{Q}_1^C$ evaluated in handle 1 of each articulated body
- $2 \times$ `Vector3d` – articulated momenta vectors $\overline{\mathbf{Q}}_1^A, \overline{\mathbf{Q}}_2^A$
- `Matrix3d` – a shift matrix $\mathbf{S}_{12}^C$ connecting handles 1 and 2 of the whole articulated body that the object represents
- $2 \times$ `Assembly`* – a pointer to the parent assemblies of the object. In the case of leaf bodies it is equal to `nullptr`.

It is worth to point out, that the program has been hardcoded as a planar MBS; hence, the dimensions of 6×6 matrices and 6-vectors reduce to 3×3 and 3-dimensional vectors (implied by the type name). The variables that are unknown apriori, such as reaction impulses or forces, are zero-initialized and assigned appropriate value later. To make use of the hierarchic structure of the DCA, we define the following variable representing the topology of the MBS: `std::vector<std::vector<Assembly>> Tree`. The variable `Tree[0]` (i.e. initial *branch* of the *Tree*) represents physical leaf-level bodies, the variable `Tree[1]` (first branch) is a vector of abstract articulated bodies computed from the branch above, etc.

Consequently, the size of outer vector (i.e., the `Tree` object) is equal to the number of branches (or `tiers`) in the binary tree, hence: `Ntiers` $=$ ceil$(\log_2(N_b)) + 1$. The size of the inner vector varies with the branch of the tree, e.g., the zeroth branch, `Tree[0]`, has a size N_b, the first branch has a size of ceil$(\frac{N_b}{2})$, and the last branch consists of a single `Assembly` object.

The code is parallelized via OpenMP directives, where three distinct parallelization mechanisms have been employed. Here, we briefly describe each approach and present an abbreviated form that appears in pseudocodes later in this section:

- `#pragma omp parallel for`: the most rudimentary OpenMP parallelization technique, where a task assigned to a `for` loop is evenly redistributed to all available threads. Abbr.: `#omp for`

- Parallel `std::vector` copy construct [21], where each thread receives a number of `std::vectors` to populate from a prescribed `for` loop. At the end of the procedure, vectors from all threads are copied (in parallel) to a global container, e.g., a `Tree` object. Abbr.: `#omp-cv for` (*copy vector*)

- Parallel cumulative sum [100], where each subsequent element of the array is the sum of its previous components. This concept is used to compute "descendant terms" in eq. (3.58), where the order of computations is reversed (from the end to the beginning). Abbr.: `#omp-cs for`

The benchmarking will be focused on a single iteration of the optimization procedure, as described by algorithm 1. The input is a fixed vector $\mathbf{b}^{\text{iter}} = \mathbf{0}$ corresponding to the control signal $u_0(t) \equiv 0$. The goal is to solve the forward and adjoint systems to compute the gradient of the performance measure. The performance of algorithm 1 will be timed and executed for different problem sizes (denoted by `Nbodies`) and numbers of threads (`Nthreads`). The timers, marked on the right side of the procedure, are started (tic) before executing the appropriate instruction and stopped (toc) right after completing the guarded line. Hence, the timer `T` will be used to measure the total time of the execution for different values of `Nbodies` and `Nthreads`. Furthermore, specific timers will tell us how long it takes to compute each logical section of the code.

Algorithm 1 Single iteration of the optimization procedure

Inputs: $\mathbf{b}^{\text{iter}}$, `Nbodies, Nthreads, input`
Declare global variables: `u, Nbodies, Nthreads, Ntiers, solnFWD, input` (including geometry, masses, simulation coefficients)

Procedure:

1: $\mathbf{u} \leftarrow$ `computeControlSignal(`$\mathbf{b}^{\text{iter}}$`)` ▷ `T`: Tic
2: `solnFWD` $\leftarrow$ `odeintRK_forwardSolver(`$t, \boldsymbol{\alpha}^0, \widehat{\mathbf{p}}^0$`)` (alg. 2) ▷ `Tfwd`: Tic/Toc
3: $\boldsymbol{\xi}^0, \boldsymbol{\eta}^0 \leftarrow$ `computeBoundaryConditions(`$\boldsymbol{\alpha}(t_f), \widehat{\mathbf{p}}(t_f)$`)` (4.13) ▷ `Tbc` : Tic/Toc
4: `solnADJ` $\leftarrow$ `odeintRK_adjointSolver(`$t, \boldsymbol{\xi}^0, \boldsymbol{\eta}^0$`)` (alg. 3) ▷ `Tadj`: Tic/Toc
5: $\nabla J \leftarrow$ `calculateGradient(solnFWD, solnADJ)` (4.16) ▷ `T`: Toc

Output: gradient $\nabla_{\mathbf{b}} J$ of the performance measure (4.1)

Algorithms 2 and 3 (called in lines 2 and 4 of alg. 1, respectively) present a detailed step-by-step description of how the contents of sections 3.3 and 4.5 have been implemented. Certain fragments of alg. 2 have been abbreviated for clarity, and the procedure describing additional DCA recursions required to compute articulated forces, accelerations, and reaction loads has been delegated to the appendix.
The instructions that begin with `odeintRK_` (lines 2 and 4 of alg. 1) are calls to odeint library, which employs RK54 (Cash-Karp) variable-step integrator with error control. The

Algorithm 2 Evaluation of the RHS of the forward problem of dynamics (HDCA)

Inputs: $t_k, \boldsymbol{\alpha}^k, \widehat{\mathbf{p}}^k$
Global variables: **u**, `Nbodies`, `Nthreads`, `Ntiers`, `input`
Numerical procedure:

$u^k \leftarrow$ `interpolateControl(`$t_k, \mathbf{u}$`)`
`Tree` $\leftarrow$ `declareDataStructure(Nbodies)`
`#omp-cv for(i=0; i < Nbodies; ++i)` ▷ `TfwdLeaves`: Tic
  `Tree[0]` $\leftarrow$ `computeLeafBodiesCoeffs(`$\boldsymbol{\alpha}^k, \widehat{\mathbf{p}}^k, u^k$`, input)` (4.49)
`end #omp-cv for` ▷ `TfwdLeaves`: Toc
Recursion from leaves to root node
`for (i=1; i < Ntiers; ++i)` ▷ `TfwdAsmVel`: Tic
  `upBranch` $\leftarrow$ `Tree[i-1]  /* upperBranch */`
  `assembliesInBranch` $\leftarrow$ `readNoAssemblies(i, Nbodies)`
  `#omp-cv for(j=0; j < assembliesInBranch; j+=2)`
    `Tree[i]` $\leftarrow$ `assembleVelocity(upBranch[j], upBranch[j+1])` (3.42)
  `end #omp-cv for`
  `if (assembliesInBranch is odd)`
    `Tree[i][end]` $\leftarrow$ `upperBranch[end]`
`end for` ▷ `TfwdAsmVel`: Toc
Backward recursion from root node to leaf bodies
`Tree[end][0].connectBaseBody()` (3.44)
`for (i=Ntiers-2; i > 0; --i)` ▷ `TfwdDisVel`: Tic
  `assembliesInBranch` $\leftarrow$ `readNoAssemblies(i, Nbodies)`
  `#omp for(j=0; j < assembliesInBranch; ++j)`
    `Tree[i][j].disassemble()` (3.39)
  `end #omp for`
  `if (assembliesInBranch is odd)`
    `Tree[i-1][end]` $\leftarrow$ `Tree[i][end]`
`end for` ▷ `TfwdDisVel`: Toc
Variable projection (joint velocity)
`#omp for (i=0; i < Nbodies; ++i)` ▷ `TfwdProj`: Tic
  $\mathbf{V}_B^1, \mathbf{V}_A^2 \leftarrow$ `evaluateAbsVelocity(Tree[0][i])` (3.31,3.32)
  $\mathbf{H}_i \leftarrow$ `computeH(`α_i`, input)` (tab. 3.2)
  $\dot{\alpha}_i^k \leftarrow$ `projectVelocity(`$\mathbf{V}_B^1, \mathbf{V}_A^2, \mathbf{H}_i$`)` (3.45)
`end #omp for` ▷ `TfwdProj`: Toc
$\overline{\mathbf{P}}_1, \overline{\mathbf{P}}_2 \leftarrow$ `computeArticulatedP(`$\widehat{\mathbf{p}}, \mathbf{H}, \mathbf{T}$`)` (3.22, 3.39)
$\overline{\mathbf{Q}}_1 \leftarrow$ `computeArticulatedForces(`$\boldsymbol{\alpha}^k, \dot{\boldsymbol{\alpha}}^k, u^k$`, Tree, input)` (alg. 4)
Descendant terms calculation
`#omp-cs for(i=Nbodies-2; i >= 0; --i)` ▷ `TfwdDes`: Tic
  $\boldsymbol{\kappa}_i \leftarrow$ `computeDescendantTerms(`$\dot{\mathbf{S}}_{C2}^i, \dot{\mathbf{S}}_{1C}^{i+1}, \overline{\mathbf{P}}_2^i$`)` (3.58)
`end #omp-cs for` ▷ `TfwdDes`: Toc
Variable projection (derivatives of joint momenta)
`#omp for(i=0; i < Nbodies; ++i)` ▷ `TfwdProj`: Tic
  $\dot{\widehat{\mathrm{p}}}_i^k \leftarrow$ `projectMomentaDerivatives(`$\mathbf{H}_1^i, \overline{\mathbf{Q}}_1^i, \boldsymbol{\kappa}_i, \overline{\mathbf{P}}_1^i$`)` (3.59)
`end #omp for` ▷ `TfwdProj`: Toc
`solnFwd` $\leftarrow \ddot{\boldsymbol{\alpha}}^k, \boldsymbol{\lambda}^k \leftarrow$ `computeAcc(`$\boldsymbol{\alpha}^k, \dot{\boldsymbol{\alpha}}^k, \overline{\mathbf{P}}_1, \overline{\mathbf{Q}}_1$`, Tree)` (alg. 5)

Output: The derivatives $\dot{\widehat{\mathbf{p}}}_i^k, \dot{\boldsymbol{\alpha}}^k$ at the time interval t_k

Algorithm 3 Evaluation of the right-hand side of the adjoint system (HADCA)

Inputs: $t_k, \widehat{\boldsymbol{\xi}}^k, \widehat{\boldsymbol{\eta}}^k$
Global variables: `u, solnFWD, Nbodies, Nthreads, Ntiers, input`
Set: in equations (3.39), (3.42): $\widehat{\boldsymbol{\xi}}_j := \mathbf{W}_j(\boldsymbol{\zeta}_{10}^B - \boldsymbol{\zeta}_{20}^A - \widehat{\boldsymbol{\xi}}_{\text{bias}})$ (4.40)
Numerical procedure:

$u^k \leftarrow$ `interpolateControl(`t_k`, u)` ▷ `TadjInterp`: Tic
$\boldsymbol{\alpha}^k, \dot{\boldsymbol{\alpha}}^k, \ddot{\boldsymbol{\alpha}}^k, \boldsymbol{\lambda}^k \leftarrow$ `interpolateState(`t_k`, solnFWD)`
$\mathbf{q}^k, \mathbf{v}^k, \dot{\mathbf{v}}^k \leftarrow$ `mapToAbsoluteVariables(`$\boldsymbol{\alpha}^k, \dot{\boldsymbol{\alpha}}^k, \ddot{\boldsymbol{\alpha}}^k$`)` (3.20), (3.21)
$\boldsymbol{\eta}^k, \boldsymbol{\xi}^k \leftarrow$ `mapToAbsoluteVariables(`$\widehat{\boldsymbol{\xi}}^k, \widehat{\boldsymbol{\eta}}^k$`)` (4.22) ▷ `TadjInterp`: Toc

Leaf bodies' coefficients calculations

```
Tree ← declareDataStructure(Nbodies)
#omp-cv for(i=0; i < Nbodies; ++i)                          ▷ TadjLeaves: Tic
  Tree[0] ← computeLeafBodiesCoeffs(q^k, v^k, v̇^k, λ^k, η^k, ξ^k, u^k) (4.53)
end #omp-cv for                                             ▷ TadjLeaves: Toc
```

Recursion from leaves to root node

```
for (i=1; i < Ntiers; ++i)                                  ▷ TadjAsm: Tic
  upBranch ← Tree[i-1]  /* upperBranch */
  assembliesInBranch ← readNoAssemblies(i, Nbodies)
  #omp-cv for(j=0; j < assembliesInBranch; j+=2)
    Tree[i] ← assembleEtaDot(upBranch[j], upBranch[j+1]) (3.42)
  end #omp-cv for
  if (assembliesInBranch is odd)
    Tree[i][end] ← upperBranch[end]
end for                                                     ▷ TadjAsm: Toc
```

Backward recursion from root node to leaf bodies

```
Tree[end][0].connectBaseBody() (3.44)
for (i=Ntiers-2; i > 0; --i)                                ▷ TadjDis: Tic
  assembliesInBranch ← readNoAssemblies(i, Nbodies)
  #omp for(j=0; j < assembliesInBranch; ++j)
    Tree[i][j].disassemble() (3.39)
  end #omp for
  if (assembliesInBranch is odd)
    Tree[i-1][end] ← Tree[i][end]
end for                                                     ▷ TadjDis: Toc
```

Variable projection

```
#omp for (i=0; i < Nbodies; ++i)                            ▷ TadjProj: Tic
  η̇¹_B, η̇²_A ← evaluateAbsEtaDot(Tree[0][i]) (4.42,4.43)
  H_i, Ḣ_i, Ḧ_i ← computeH(α_i, input)
  ξ̂_bias ← computeCBias(η̂^k, ξ̂^k, Ḣ_i, Ḧ_i) (4.40), (tab. 4.3)
  η̇̂^k_i ← projectVelocity(η̇¹_B, η̇²_A, H_i, ξ̂_bias) (4.54)
end #omp for                                                ▷ TadjProj: Toc
ξ̇̂^k ← η̂^k
```

Output: The derivatives $\dot{\widehat{\boldsymbol{\eta}}}^k, \dot{\widehat{\boldsymbol{\xi}}}^k$ at the time interval t_k

integration routine performs multiple calls to the appropriate right-hand side function defined by algorithms 2 and 3. The numbering convention within the arrays is as follows: the first element of the array starts with index $i = 0$, and the last element is denoted by `end`.

6.3 Results validation

Before testing the efficiency, let us verify the correctness of the developed formulation. To this end, we compare the implementation results of alg. 2 with the outcome recorded by commercial multibody solver MSC ADAMS. Default simulation parameters were used (GSTIFF integrator, index-3 formulation) with error tolerance reduced to 10^{-6}, while the parameters of the multibody model are gathered in table 5.3 (here, $t_f = 2$ s). The compared case assumes $N_b = 4$, and figure 6.2 displays the dynamic response from both models. Plots show $x, \dot{x}$ coordinates of the first body (in meters and $\frac{\mathrm{m}}{\mathrm{s}}$) and $\varphi, \dot{\varphi}$ coordinates (orientation in radians and angular velocity in $\frac{\mathrm{rad}}{\mathrm{s}}$) of the last body. Since the plots look identically, we also present the calculated difference between corresponding plots in fig. 6.3.

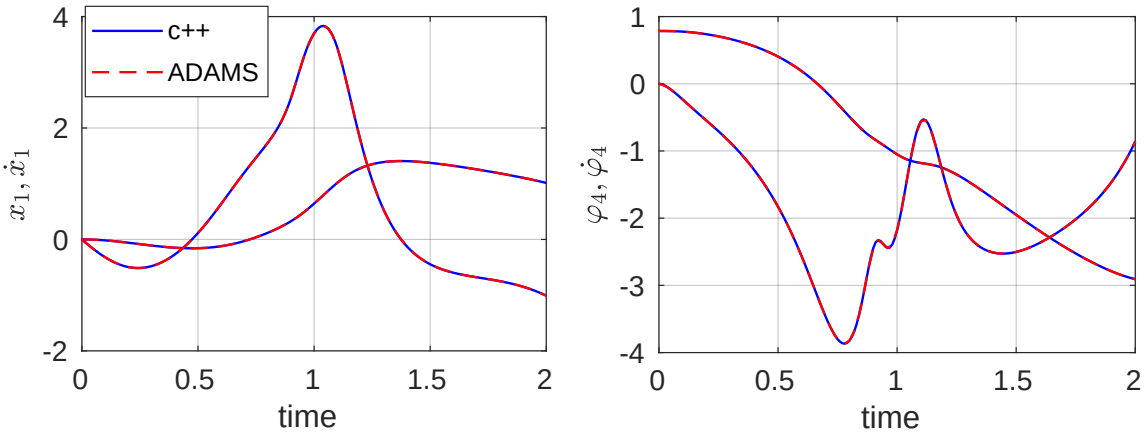


Figure 6.2: The dynamic response produced by custom C++ solver and ADAMS software

Consequently, the implementation of the procedure 3, which solves the adjoint system, must also be validated. To this end, we compare the implementation results with the outcome of a corresponding global formulation of eq. (4.14), which has been validated in multiple test cases, such as the ones presented in chapter 5 or refs. [79, 81]. According to eq. (4.16), the adjoint variable ξ_1 maps trivially to the gradient ∇J since $h_{\mathbf{b}} = \mathbf{0}$ and $\mathbf{f}_{\mathbf{b}} = [1, \mathbf{0}]$. These relations result, accordingly, from eq. 5.5 and the fact, that $u(t)$ acts directly on the first body, i.e. $\mathbf{f}(t) = [u(t), \mathbf{0}]^T$. Therefore, ξ_1 can be treated as the

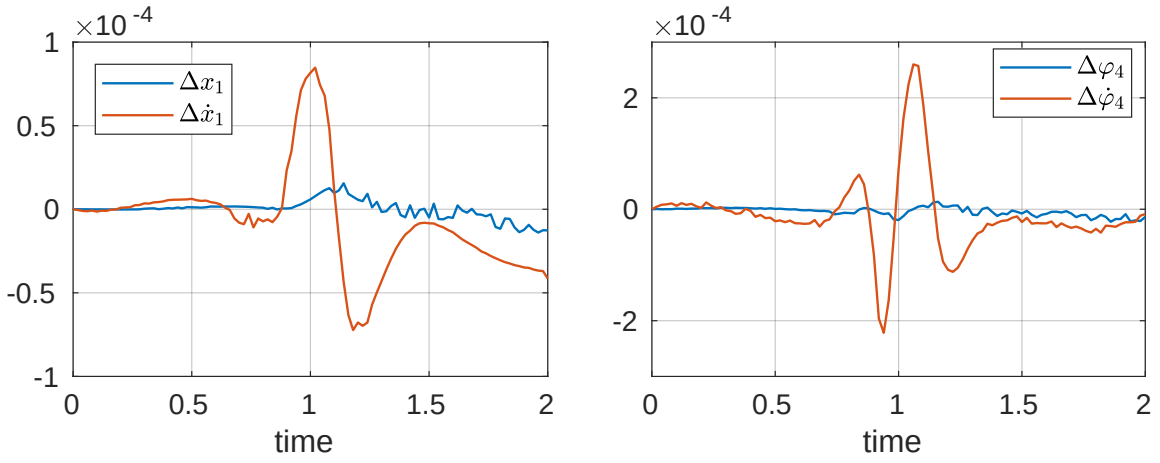


Figure 6.3: Comparative results between the outcome produced by custom C++ solver and ADAMS package

result of the adjoint sensitivity analysis. In the following example, the result of tested HADCA approach, $\widehat{\xi}_1(t)$, is equivalent with its absolute-coordinate counterpart since $\widehat{\xi}_1 = \mathbf{H}_1^T \boldsymbol{\xi}_1 = [1,0,0] \cdot [\xi_1, \xi_2, \xi_3]^T = \xi_1$. Figure 6.4 confirms that both methods' outcomes are nearly identical, as predicted. Furthermore, figure 6.5 compares the residual values of algebraic constraints between both formulations (c.f. tab. 4.2). The developed HADCA approach fulfills all equations with machine accuracy since only joint-space variables are integrated (leading to ODEs). On the other hand, eq. (4.14) is a set of DAEs that explicitly enforces only the equation $\ddot{\boldsymbol{\Psi}} = \mathbf{0}$. The lower-order residual functions behave exactly as expected when no stabilization technique is employed.

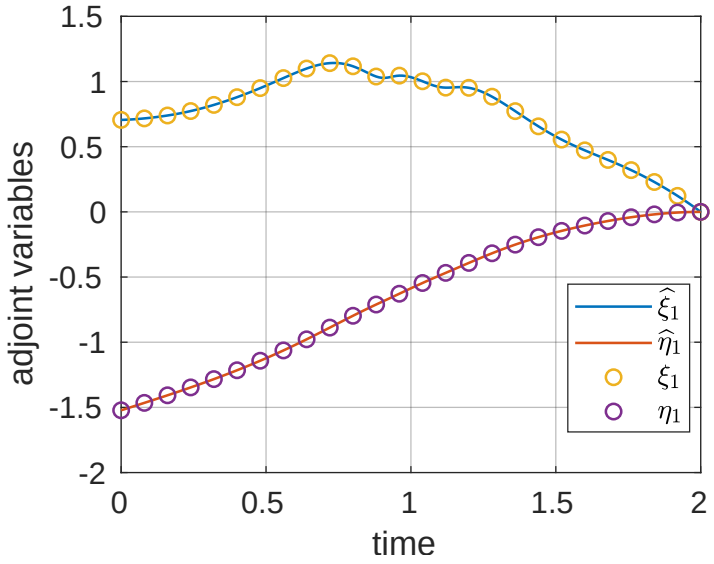


Figure 6.4: Comparison of resultant adjoint variables for two approaches: tested HADCA ($\widehat{\xi}_1, \widehat{\eta}_1 = \dot{\widehat{\xi}}_1$) and global ($\xi_1, \eta_1 = \dot{\xi}_1$)

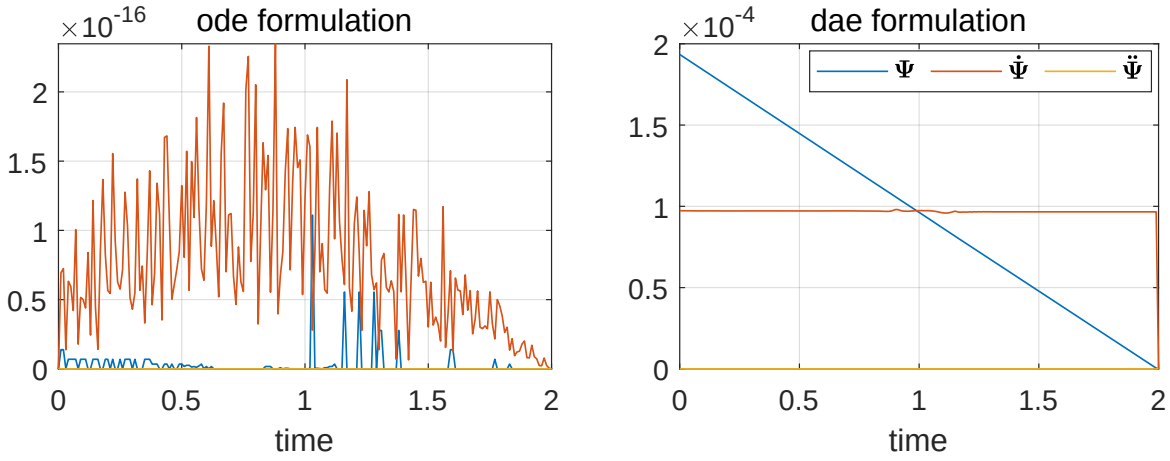


Figure 6.5: The residual values of algebraic constraints for two approaches: tested HADCA (left) and global (right) formulation

6.4 Scalability Analysis

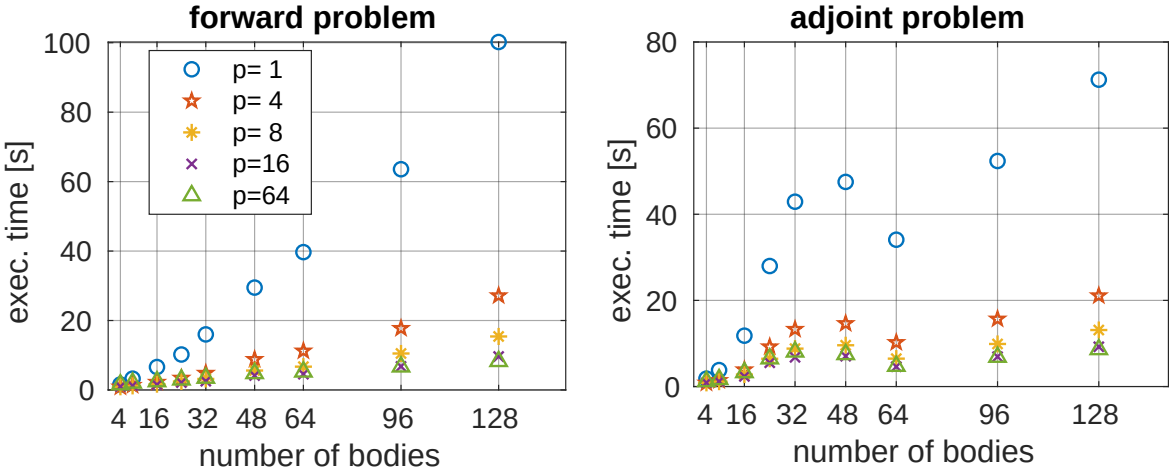


Figure 6.6: Execution times of the algorithm 1 for different problem sizes and different number of available threads

Figure 6.6 presents the results of multiple runs varied by the number of available threads (marked by different data series) and the problem size displayed on the horizontal axis. The measurements treat the forward and adjoint problems separately. The execution of the adjoint problem reveals certain memory-specific issues manifested at problem sizes equal to $N_b = 36$ or $N_b = 48$, where the performance becomes slower than for the problem size of $N_b = 64$ bodies. Although this behavior is visible for all values of `Nthread`, the general pattern for small and medium scales significantly boosts computations for cases with up to $p = 16$ available threads. Another way to visualize gathered measurements is to present a speedup of the executed code in relation to a single thread execution.

Specifically, the speedup coefficient S_p for a problem size N_b and achieved by paralleling p threads is calculated as:

$$S_p = \frac{T_1(N_b)}{T_p(N_b)},$$

where T_p is the execution time for p threads. Figure 6.7 depicts the speedup for three most effective cases, namely $\texttt{Nthread} = p = \{4, 8, 16\}$.

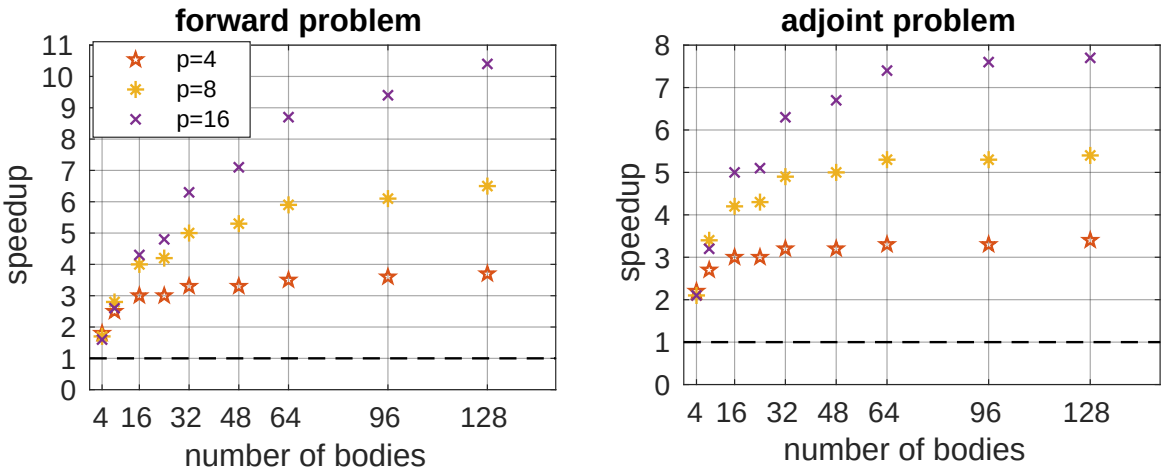


Figure 6.7: Speedup in relation to a single thread execution for different problem sizes and specified number of available threads

Furthermore, certain data points, specifically $N_b = \{32, 128\}$ and $p = \{1, 32\}$, were picked for a more detailed analysis. The timers defined on the right side of the pseudocode 2 and 3 have been employed to measure the execution times of each logical section of the implementation. The results are displayed in figure 6.8 and will be discussed in the remainder of this subsection.

Each bar plot refers to the specific dataset and is divided into forward and adjoint problems. The ordinate axis describes a fraction of the total time consumed by a section of code, and the total time (indicating 100%) is denoted in the title of each subplot. The first five labels of the legend refer to both forward and adjoint problems, e.g., the label *Leaves* refers to `TfwdLeaves` (alg. 2) and `TadjLeaves` (alg. 3), respectively. The label *misc* (miscellaneous) is the difference between the external timers `Tfwd, Tadj` (c.f. alg. 1) and the sum of all internal timers in algorithms 2, 3. Hence, it involves the fragments of code, that have not been captured by any of the specified timers, e.g., type casts between variables, saving data in the workspace, etc. Finally, the *subset* label is the sum of six timers defined in algorithms 4 and 5, i.e., it is the duration of two additional assembly-disassembly recursions required to compute forcing and accelerations terms.
We can draw certain conclusions from figure 6.8, specifically:

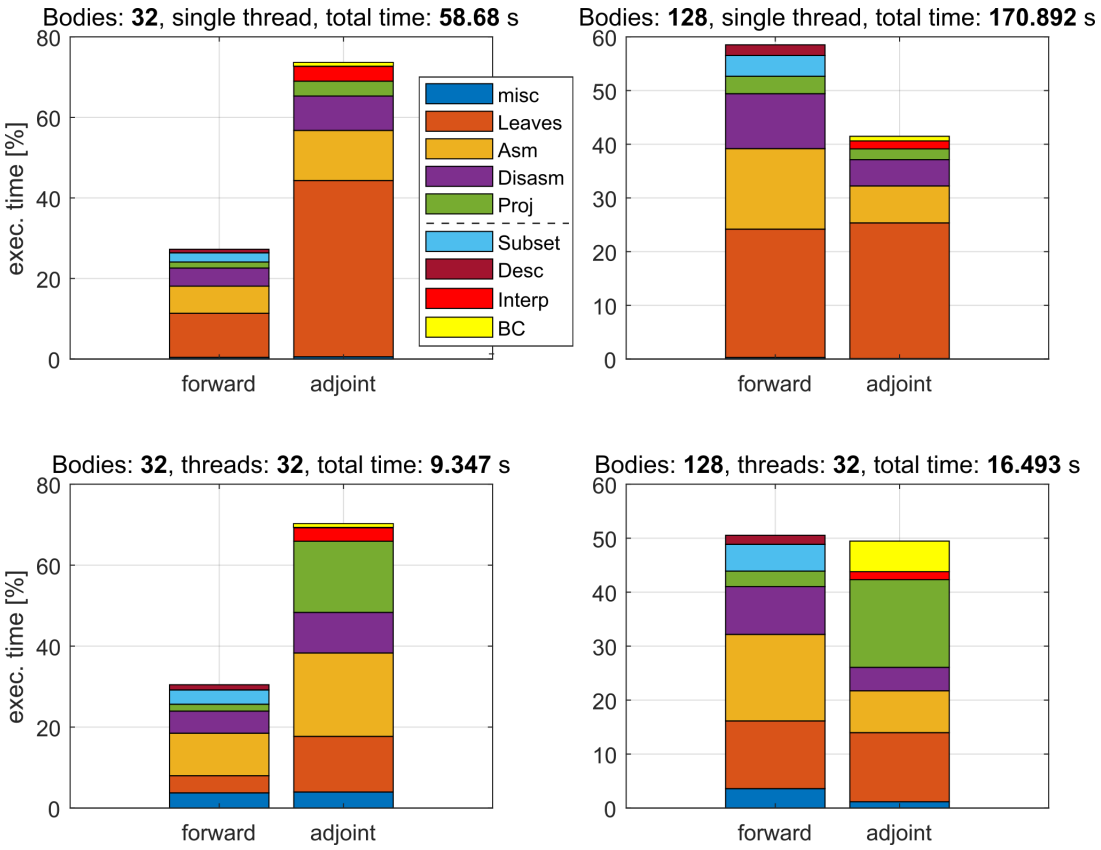


Figure 6.8: Detailed execution measurements of different logical sections of the implementation for various data sets

1. Horizontal comparison: the overall time required to solve the problem nearly tripled in the case of a single threaded execution (c.f. upper plots), whereas in the latter case (lower plots, 32 threads), this duration has been less than doubled. The computational intensity rose with the problem size, and consequently, the efficiency of the parallel implementation increased as well, as indicated by fig. 6.7. Furthermore, the relative share of the time consumed for solving the adjoint problem shrank with the number of bodies in the system.

2. Vertical comparison: the parallelization affected computations performed over the initial branch of the binary tree (i.e., *Leaves*) to the greatest extent. This behavior is expected since *Leaves* is the most concurrent part of the algorithm (c.f. point 3). On the other hand, the relative execution of a section of code associated with the projection of variables extended almost five times with respect to single-threaded execution.

3. The most time-consuming section of the procedure leans on establishing the EOM's initial, leaf-body level coefficients, critically in the case of single thread computations and the adjoint problem. Fortunately, the leaf-body level is the most concur-

rent (i.e., parallelizable) section of the algorithm. As a result, the parallelization leads to more balanced distribution of the workload, as verified by observations. Potentially, this is the place where one can gain additional benefits from software optimization involving hardware-specific features such as Intel's AVX (Advanced Vector Extensions) or proper cache management.

4. The same coefficient matrix in eqs. (3.14) and (4.14) allows to reuse coefficients $\zeta_{11}, \cdots, \zeta_{22}$ computed in the forward problem again in the adjoint problem. This approach is feasible when a fixed-step integrator is employed since, otherwise, additional interpolation of these coefficients must be performed. Conversely, in the case of a variable-step integrator (employed herein), one may interpolate only $2N_b$ joint state variables and use them to recreate all other parameters. This approach manifested itself in the results in two ways: the *Interp* (i.e., data interpolation) section is relatively short, and the DCA recursion of the forward (velocity level) and adjoint problems takes a similar amount of time to execute.

5. The *Leaves* section tends to be more intensive in the case of the adjoint problem since the expressions shown in eqs. (4.53) are significantly more expensive to evaluate than eqs. (4.49).

6. In the forward problem, the single DCA recursion on the velocity level (*Leaves, Asm, Disasm*) takes significantly more time than the two following recursions described by algorithms 4, 5 (*Subset*). This behavior can be explained by a lower amount of computations in the latter case and by the specifics of the implementation: only *Leaves* and *Asm* phases involve more expensive `#omp-cv for` loop that establishes internal dependencies between the binary tree. The following recursions utilize these dependencies via the internal pointer of the `Assembly` class, which allows for the execution of the code in a simpler `#omp for` loop.

7. The boundary conditions (*BC*) of the adjoint problem become relatively more time consuming in the case $N_b = 128$, $p = 32$. This observation stems from the implementation, where boundary adjoint variables are computed directly from eq. (4.13). The cost of solving such a system with size, e.g., $3 \cdot 512 = 1536$ (3 variables per body) is relatively large. One can significantly reduce the workload by implementing another DCA recursion specifically for it. Nevertheless, current implementation presents good opportunity to pinpoint an interesting comparison regarding this case. As indicated by fig. 6.8, the duration of solving eq. (4.13) directly and only once is barely shorter than the accumulated times needed to solve eq. (4.14) via DCA for all time intervals of the integration procedure (i.e. at least $\frac{t_f}{\Delta t} = \frac{2}{0.01} = 200$ times).

6.5 Summary

This chapter presented a detailed description of a computer code, which implements methods developed in the previous chapters. Specifically, algorithm 2 synthetically described the key concepts of section 3.4.1, whereas algorithm 3 captured the details of section 4.5. To simplify the programming, the optimal control problem was hardcoded in the form of a planar multibody system with a varying number of bodies. The developed code has been tested against various aspects, such as:

- The correctness of computations for both forward and adjoint problems
- The execution time vs. problem size (i.e., the number of bodies)
- The execution time vs. the number of available threads working in parallel

Conducted benchmarking showed that the applied algorithms significantly speed up the execution of the gradient computation. Furthermore, the execution times of many logical sections within the algorithm were precisely measured, which allowed us to understand the underlying methods to the extent possible.

CHAPTER 7

EXPERIMENTS: OPTIMAL CONTROL OF A TWO-DEGREE-OF-FREEDOM ROBOT

7.1 Introduction

The problems from chap. 5 have been solved, and their outcome was verified in a simulation environment. This chapter aims to extend the developed methods' scope by implementing them on a real object. To this end, we will split this task into two complementary parts. Firstly, we are going to generate optimal trajectories with the aid of the computational model of the hardware by utilizing the adjoint method. The outcome will be used in the following step as a feedforward signal in the control architecture depicted in figure 7.1. As suggested in the diagram, the control loop involves a feedback signal as well, which in this case will constitute a PD controller. The generated output will be measured and compared with the model-free approach based on the classical PD-controller.

7.2 Control architecture

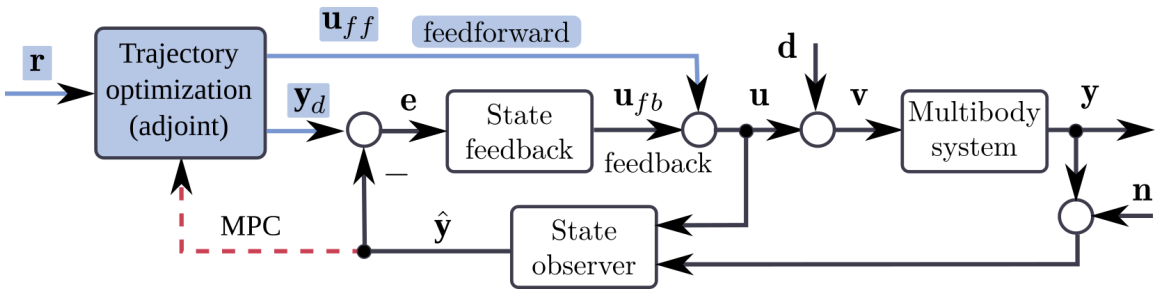


Figure 7.1: A feedback-feedforward control architecture

Figure 7.1 presents a control architecture layout that will be applied to the two-degree-of-freedom robot investigated in this chapter. The main blocks in the diagram are *multibody system*, represented by the actual hardware (or its model in the case of the offline execution), and *trajectory optimization*, characterized by the adjoint-based optimization procedure. Furthermore, the *state feedback* block illustrates a PD controller, whereas the *state observer* designates a set of sensors, specifically encoders reading the current orientation of the motor shaft.

Symbol $\mathbf{r}$ denotes a reference signal that nominally must be enforced. Accordingly, this signal plays the role of the input to the adjoint-based optimization procedure, founded on the mathematical model of the MBS. The optimization algorithm yields the control signal $\mathbf{u}_{ff}$ (theoretically) capable of carrying out the required maneuver. The response generated by the model for $\mathbf{u}_{ff}$ is depicted as $\mathbf{y}_d$, and it becomes the actual reference trajectory for the system. Due to the discrepancy between the model and the plant, as well as the presence of disturbances $\mathbf{d}$ and measurement noise $\mathbf{n}$, it is required to maintain a feedback loop that executes minor corrections $\mathbf{u}_{fb}$ during the online execution. The remaining signals appearing in the diagram can be described as follows:

- $\mathbf{y}$ – state variables of the dynamic system
- $\widehat{\mathbf{y}}$ – measured state variables
- $\mathbf{u}$ – computed input control signal
- $\mathbf{v}$ – actual (disturbed) input signal
- $\mathbf{e}$ – computed error value

The dashed red line refers to Model Predictive Control (MPC), also known as receding horizon control. It symbolizes the fact that computations take place online during the maneuver. This approach is most feasible where the controlled processes are sufficiently slow to permit its implementation. Although possible, implementing MPC in dynamic systems or robotics environments is relatively challenging [83, 93]. Herein, it has been specifically highlighted to pinpoint that the trajectory optimization is performed offline, and the MPC approach is a subject open to development.

7.3 Problem setup

The measurement circuit with the highlighted mechanism on which the experiment has been carried out is presented in figure 7.2. The multibody system constitutes a five-bar linkage actuated by two DC motors with a gear transmission. An extended variant of this

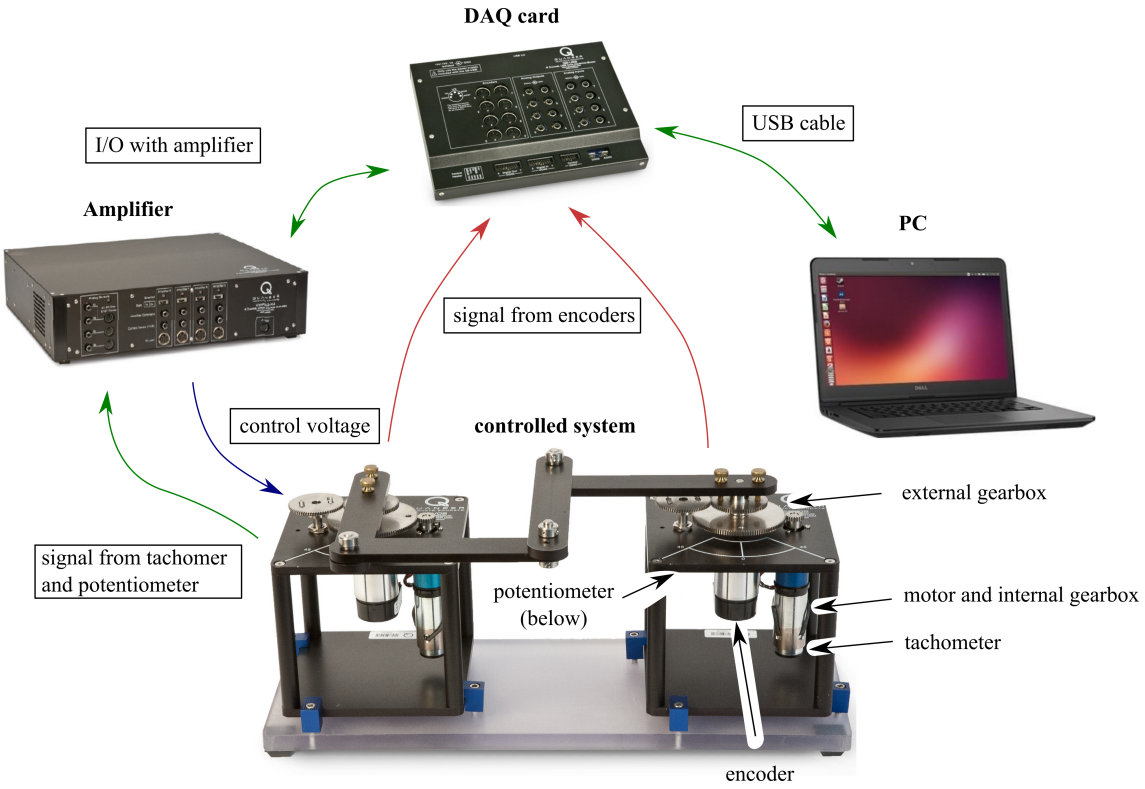


Figure 7.2: Five-bar linkage driven by two DC motors

multibody system (with a pendulum attached to the linkage) has been already presented in section 5.5. The device has a variety of sensors, such as a tachometer, potentiometer, and encoder. The motors are controlled by supplying the appropriate voltage from the data acquisition (DAQ) card and the amplifier. The communication with the device is carried out via PC with Simulink environment and an additional library known as Quarc. The Simulink block diagram is compiled into an automatically generated C code and sent to the DAQ card connected to the PC. Consequently, the DAQ card is attached to the device's sensors as well as the amplifier, which sends the required voltage signals into the DC motors.

Let us distinguish a point $\mathbf{P}$ on the five-bar linkage (c.f. fig. 5.21), which we will refer to as the end-effector of this MBS. Since figure 7.2 depicts a starting configuration of the system, and the length of each link is equal to $l = 5$ inch, the initial position of the end-effector is equal to $\mathbf{P}^{\text{init}} = [5, 5]$ inch (c.f. 5.21). The end-effector is supposed to follow a prescribed trajectory in a fixed time. Specifically, we define two Lissajous curves, denoted by the symbols $\mathcal{A}$ and $\mathcal{B}$ and characterized by the following parametric equations (expressed in inches):

$$x(t) = 5 + 2\sin(\omega_x t), \qquad y(t) = 5 + 2\sin(\omega_y t), \tag{7.1}$$

where curve $\mathcal{A}$ is determined by $\omega_x = 1.26\frac{\text{rad}}{\text{s}}, \omega_y = 2\omega_x, t_f = 5$ s, and curve $\mathcal{B}$ has the following parameters: $\omega_x = 2.5\frac{\text{rad}}{\text{s}}, \omega_y = \frac{3}{2}\omega_x, t_f = 5$ s. The final time, t_f, has been prescribed in a specific manner resulting in one full revolution of the end-effector around the workspace. Equation (7.1) describes the signal $\mathbf{r}$ in figure 7.1.

The correct execution of the specified task requires determining a sufficiently reliable electromechanical model of the multibody system presented in figure 7.2. The mathematical model of the DC motors is described by eq. (5.6) whose parameters are gathered in table 5.4. Furthermore, the dynamics are captured by Hamilton's equations of motion (3.9). To test the degree of the model's correctness, one can execute an open-loop control (instead of the scheme presented in fig. 7.1) on the hardware. The model's accuracy is verified by comparing the measured output from the device with the computed output from the simulation. The input voltages remain the same in both cases, depicted in fig. 7.3. Figure 7.4 presents the underlying trajectory of point $\mathbf{P}$ in Cartesian coordinates compared with the trajectory generated by the end-effector. Consequently, figure 7.5 compares the orientations of the actuated bodies (left plot) as well as the angular velocity of the first motor (right plot).

One can conclude from fig. 7.5 that the model output is qualitatively similar to the trajectory generated by the device. Specifically, the predicted velocity accurately overlaps with the measured signal. Nevertheless, the model significantly undershoots the actual response, which is visible on the left plot of fig. 7.5 as well as on fig. 7.4. This issue may have been caused by phenomena not included in the mathematical model, such as friction in joints, or uncertainty of the model parameters, such as electromechanical motor friction coefficients, other motor constants, or inertia parameters of the linkage. Ultimately, the results generated with the aid of a model-based procedure must rely on the feedback via, e.g., a PD controller to compensate for exposed limitations.

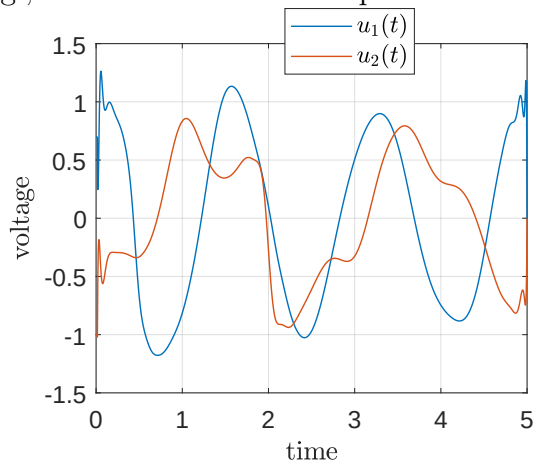


Figure 7.3: Input signals supplied to model and the hardware

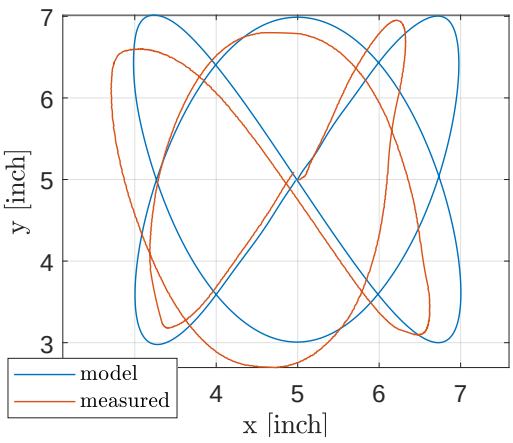


Figure 7.4: The trajectories obtained from model and measurements

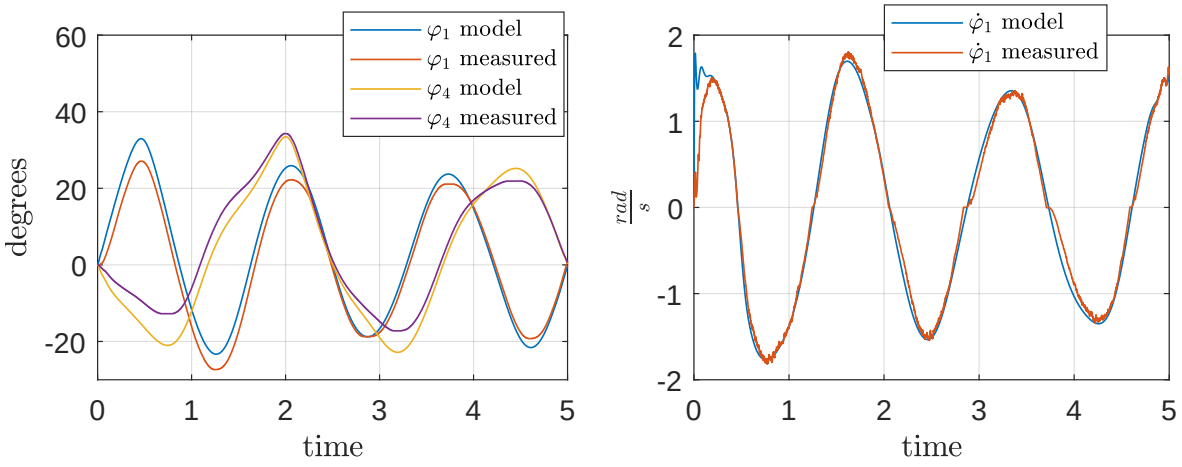


Figure 7.5: Dynamic response of the mathematical model compared with measurements obtained form the device

7.4 Feedforward and feedback signals

The classical approach to following a prescribed trajectory of the end-effector requires solving the inverse kinematics problem. Specifically, it involves computing the *internal variables* for a given configuration of the MBS, i.e., the independent variables that uniquely describe the configuration of the system [124]. In this case, the internal variables are angles φ_1, φ_4 (marked in fig. 5.21), and they are uniquely determined by the Cartesian coordinates of point $\mathbf{P}$. Classically, these angles (computed via inverse kinematics) constitute a reference for measured orientations of the actuated bodies, whereas the difference between measurements and reference is fed to the PD controller. The velocity component for the derivative action is obtained via a numerical differentiation provided by the appropriate block of the Simulink library. The following coefficients of the controller are prescribed: $k_p = 22.7$ (proportional gain), $k_i = 0.43$ (integral action gain). In this particular scenario, the reference values of the internal variables (i.e., the inverse kinematics problem) will be computed offline (i.e., before executing the trajectory on a robot) via the optimization method.

The architecture portrayed in the previous paragraph will be a reference for a more involved procedure, where the input voltages are obtained based on the mathematical model of the device. To this end, we introduce the following performance measure to be minimized:

$$J = (1-\alpha)\int_0^{t_f} \Big((\mathbf{r_P} - \mathbf{r}^*_{\mathbf{P}}(t))^T (\mathbf{r_P} - \mathbf{r}^*_{\mathbf{P}}(t)) + \beta (\dot{\mathbf{r}}_{\mathbf{P}} - \dot{\mathbf{r}}^*_{\mathbf{P}}(t))^T (\dot{\mathbf{r}}_{\mathbf{P}} - \dot{\mathbf{r}}^*_{\mathbf{P}}(t)) \Big)\, \mathrm{d}t \\ + \alpha (\mathbf{r_P} - \mathbf{r}^*_{\mathbf{P}}(t))^T (\mathbf{r_P} - \mathbf{r}^*_{\mathbf{P}}(t))|_{t_f}. \tag{7.2}$$

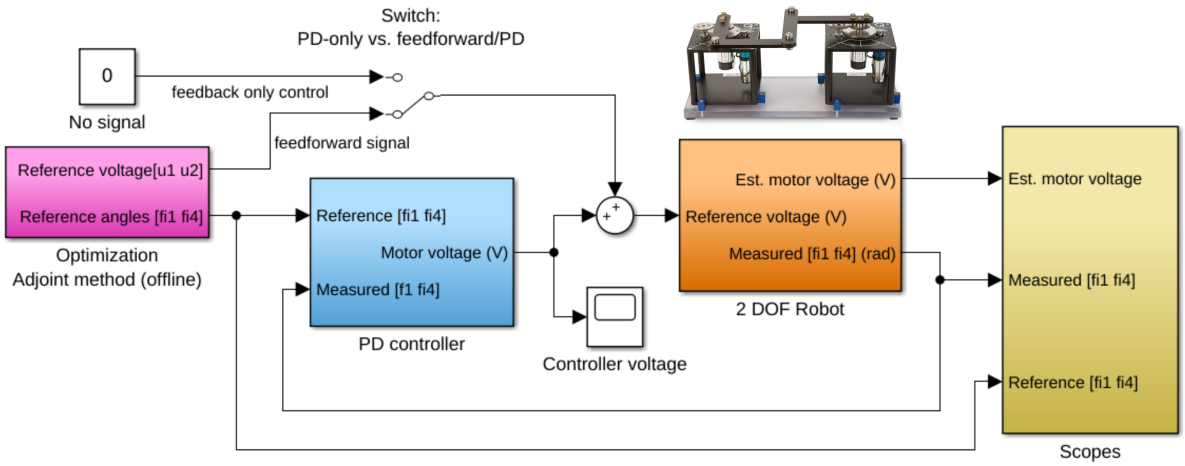


Figure 7.6: Simulink model of two investigated control architectures. The manual switch allows to set either model-based or model-free approach

Here, $\mathbf{r_P}$ refers to the computed value of point $\mathbf{P}$, whereas the symbol $\mathbf{r}^*_\mathbf{P}(t)$ represents the time-dependent trajectory described by eq. (7.1). Symbols $\alpha = 0.05$ and β are taken as weighting factors. Furthermore, the design variables are discretized input voltages $u_1(t), u_2(t)$ (c.f. sec. 5.5). Based on the mathematical model, these quantities produce corresponding trajectory $\mathbf{r_P}$ required to evaluate eq. (7.2) and internal variables φ_1, φ_4. The z coordinate remains constant and is equal to zero. Figure 7.6 depicts a block diagram that captures the classical control architecture (based only on a PD control) and a model-based control design founded on the feedforward signal computation. It is assumed that each motor is controlled via a separate PD controller, whereas the controllers are independent of each other. Tasks that involve tracking a time-dependent reference trajectory constitute a family of problems with *control* (or *servo-*) *constraints* and have been extensively covered in the literature [10, 15].

An optimization-based approach will be presented in the following part of this chapter, where the gradient information is obtained via the adjoint method introduced in chapter 4. As indicated in fig. 7.1, the solution to the optimal control problem can be inserted into the control loop as a predicted input signal together with a corresponding trajectory. The trajectory optimization problem is solved offline with the Matlab SQP algorithm in this particular scenario. For better accuracy, the optimization is divided into two consecutive tasks: both rely on minimizing performance measure (7.2); however, initially, we assume $\beta = 0$ and switch to $\beta = 1$ in the subsequent case. One can see that the initial subproblem focuses only on the position-based analysis and aims to generate a feasible initial guess for the subsequent task, in which we take into account velocity-based components to achieve more accurate results. Figures 7.7, 7.8 present the iterative history of the optimization

for the curves $\mathcal{A}$ and $\mathcal{B}$, respectively. The switching of the parameter β is denoted with dashed vertical line.

The two practical results obtained from the optimization are input signals (i.e., motor voltages) and reference internal variables that correspond to the trajectory of the end-effector predicted by the model. Figure 7.9 presents the trajectories generated with the aid of the adjoint method (dashed orange line) against the nominal trajectories described by eq. (7.1). Instead of applying inverse kinematics to equation (7.1), we utilize adjoint-based trajectory generator for φ_1, φ_4. This means that the former trajectory will constitute a reference signal for the control architectures in the remainder of this chapter. Last but not least, figure 7.10 shows a snapshot of the device taken while realizing trajectory $\mathcal{B}$.

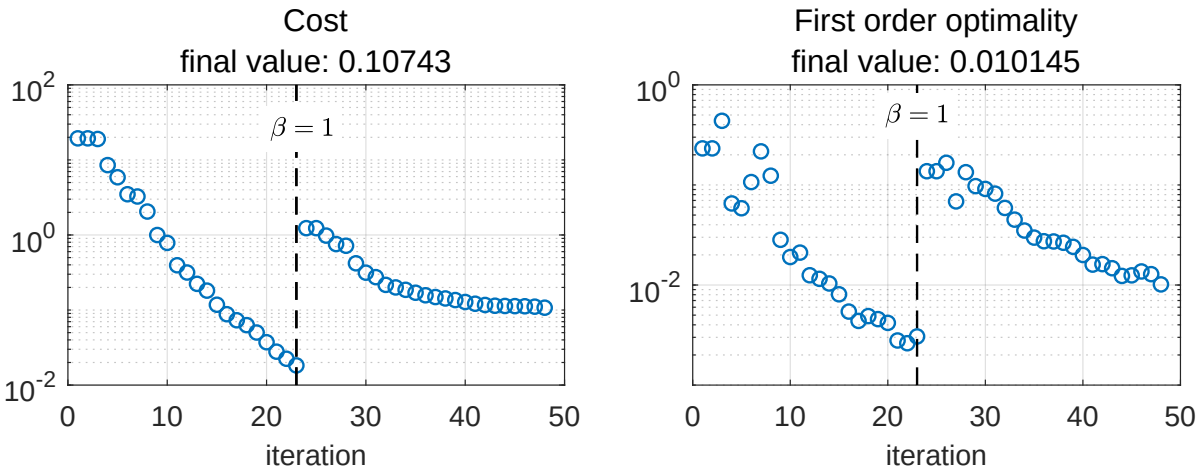


Figure 7.7: Optimization history for the model of a five-bar linkage following trajectory $\mathcal{A}$ (c.f. eq. (7.1))

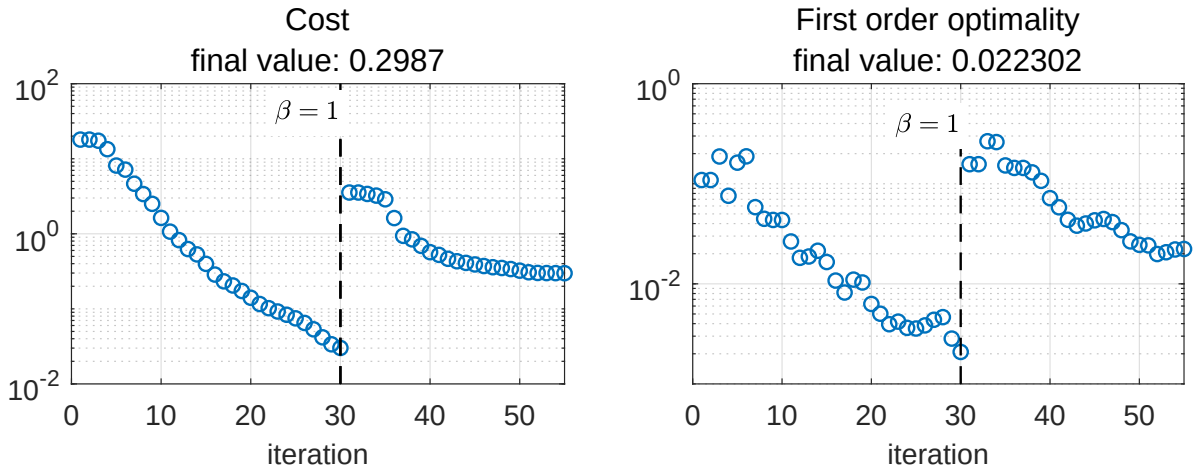
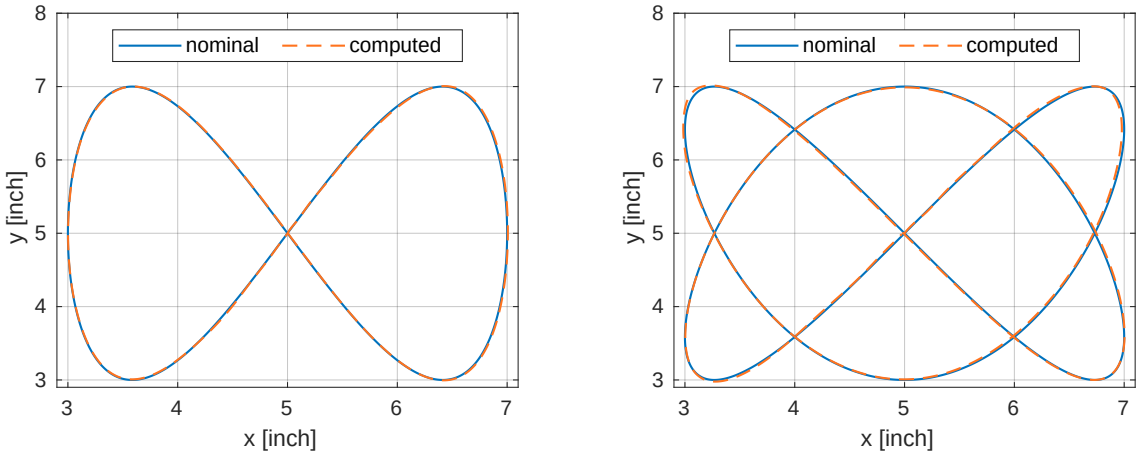


Figure 7.8: Optimization history for the model of a five-bar linkage following trajectory $\mathcal{B}$ (c.f. eq. (7.1))



(a) Trajectory $\mathcal{A}$ (b) Trajectory $\mathcal{B}$

Figure 7.9: Nominal (generated with eq. (7.1)) and computed trajectories of the five-bar's end-effector

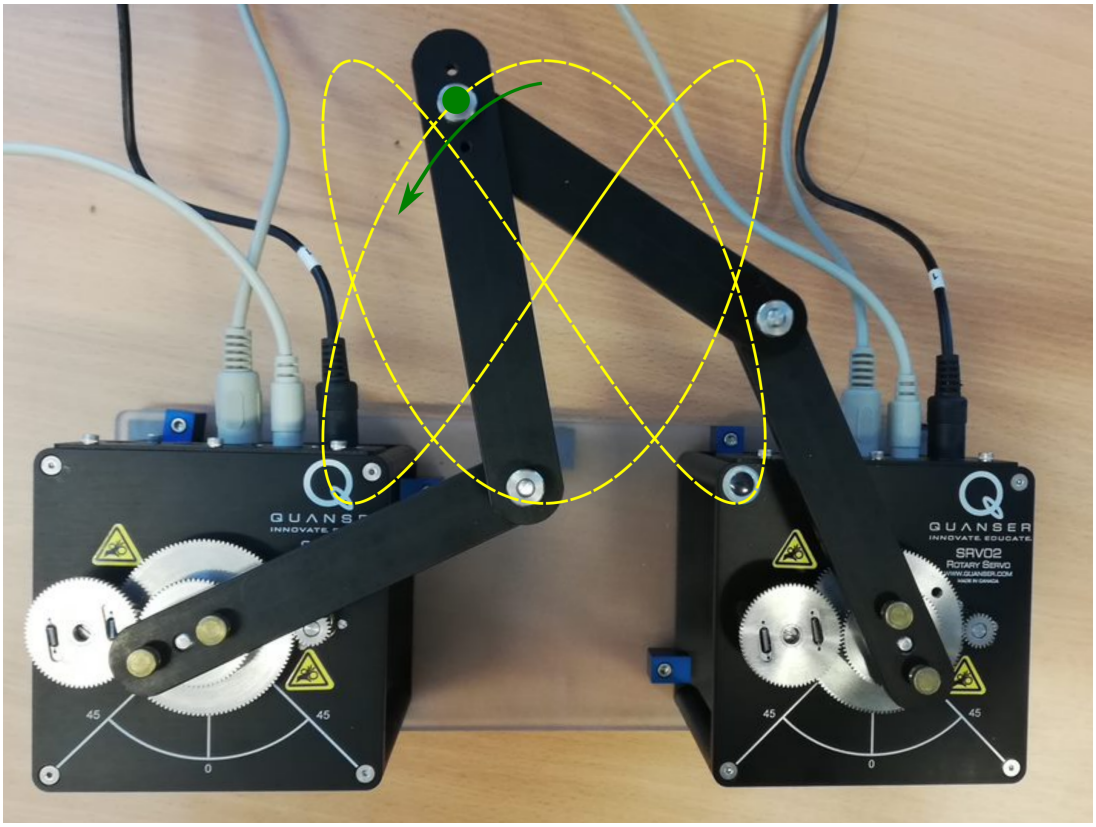
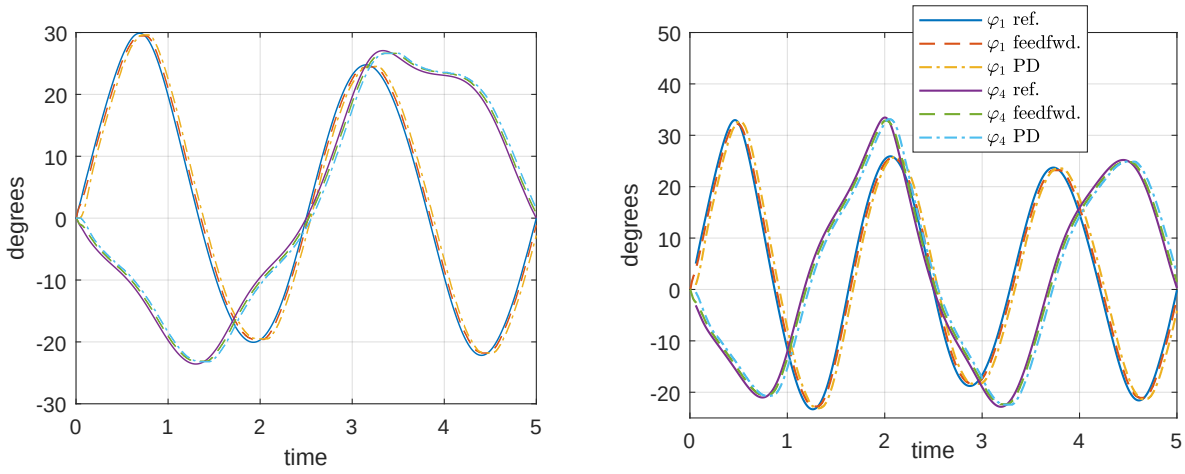


Figure 7.10: A top-down snapshot of the device during trajectory realization

7.5 Results



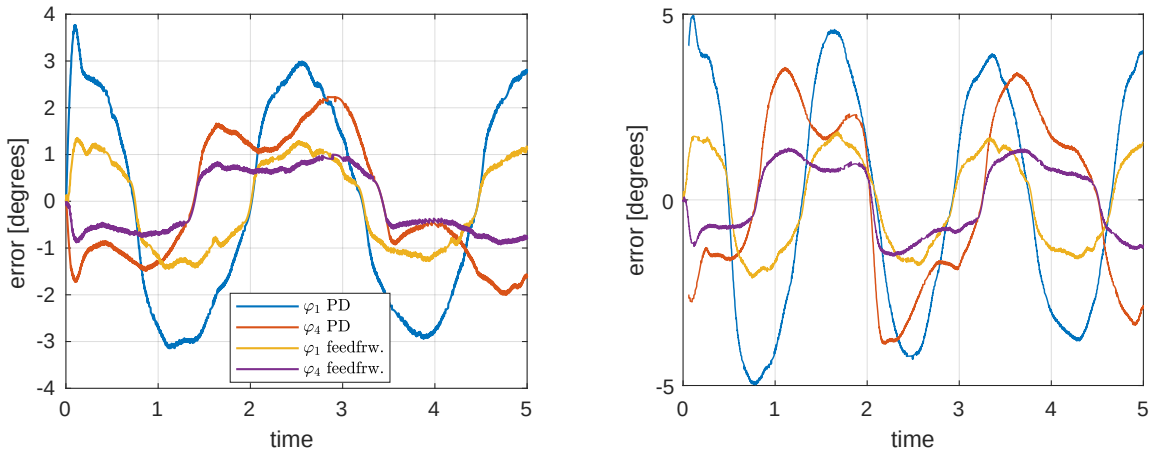
(a) Trajectory $\mathcal{A}$ (b) Trajectory $\mathcal{B}$

Figure 7.11: Internal angles measured for feedback-only and feedforward/feedback executions compared with their reference values

Since the mathematical model is a mere approximation of the real dynamic system, a proper execution of the open-loop (adjoint-based) control algorithm still requires application of a feedback loop. The proportional and derivative gains remain the same in both model-based and model-free (i.e., based only on PD control) setups. Various trajectories have been executed on the tested device, and this section summarizes the results and obtained measurements regarding curves $\mathcal{A}$ and $\mathcal{B}$.

As stated earlier, the end-effector must follow the reference trajectory generated by the optimization procedure backed up by the adjoint method. Figure 7.11 presents the measured response of the internal angles for executions with PD-only as well as feedback + feedforward setups. The solid lines represent reference angles. Since both approaches track the reference reasonably well, it is worth comparing the error, i.e., the deviation from the reference. This measure is shown in figure 7.12, from which one can observe that applying a feedforward signal to the control loop significantly reduces the tracking error of both variables.

Furthermore, it is possible to compute the path of point **P** by feeding measured variables to the forward kinematics problem. Hence, we can graph a recreated trajectory expressed in Cartesian coordinates to visually compare the results. Figure 7.13 presents trajectories generated by PD-only control architecture, whereas fig. 7.14 depicts results of a model-based approach. Additionally, we can quantify the discrepancy between a reference trajectory ($\mathbf{r}_{\mathbf{P}}^{\text{ref}}$) and the measured outcome ($\mathbf{r}_{\mathbf{P}}^{\text{meas}}$) by computing the integral squared error (ISE) measure:

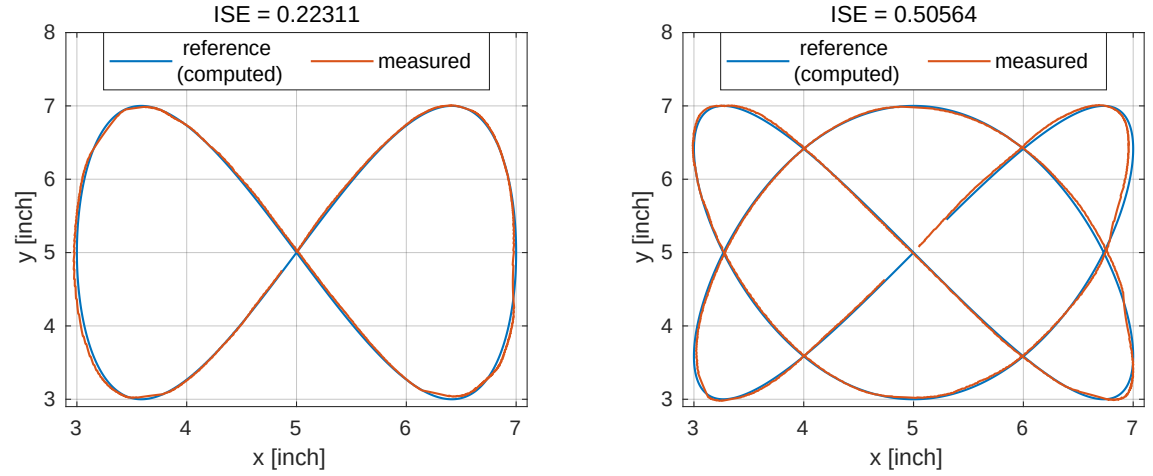


(a) Trajectory $\mathcal{A}$ (b) Trajectory $\mathcal{B}$

Figure 7.12: Error between measured angles and the reference values recorded for two executions (PD-only and feedforward+PD)

$$ISE = \int_0^{t_f} (\mathbf{r}_\mathbf{P}^{\text{meas}} - \mathbf{r}_\mathbf{P}^{\text{ref}})^T (\mathbf{r}_\mathbf{P}^{\text{meas}} - \mathbf{r}_\mathbf{P}^{\text{ref}})\, \mathrm{d}t. \tag{7.3}$$

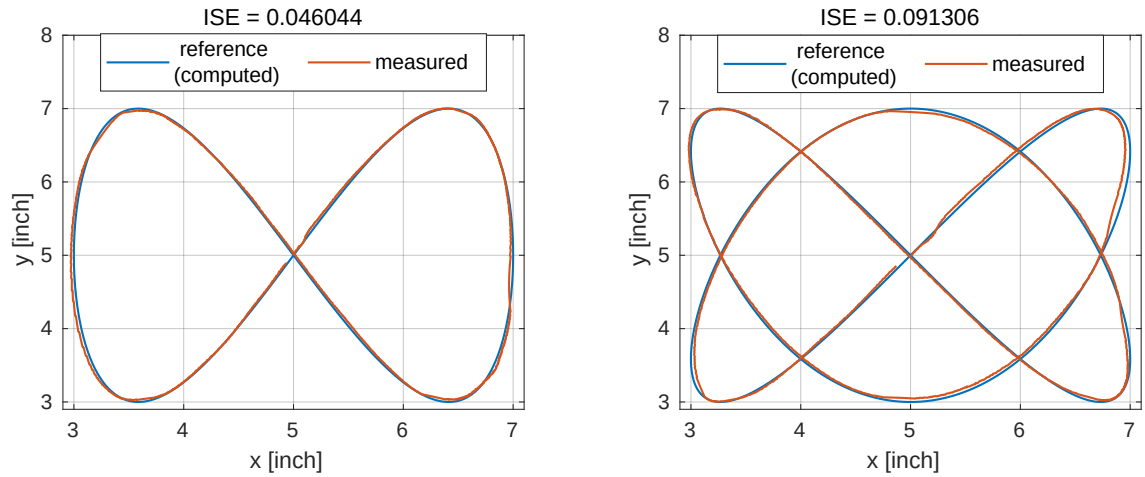
Its value is marked at the top of each figure, and we can observe that the tracking measure is an order of magnitude smaller in cases, where the feedforward signal is fed to the control loop.



(a) Trajectory $\mathcal{A}$ (b) Trajectory $\mathcal{B}$

Figure 7.13: Cartesian coordinates representation of the recreated trajectory compared with the reference for the PD-only control architecture

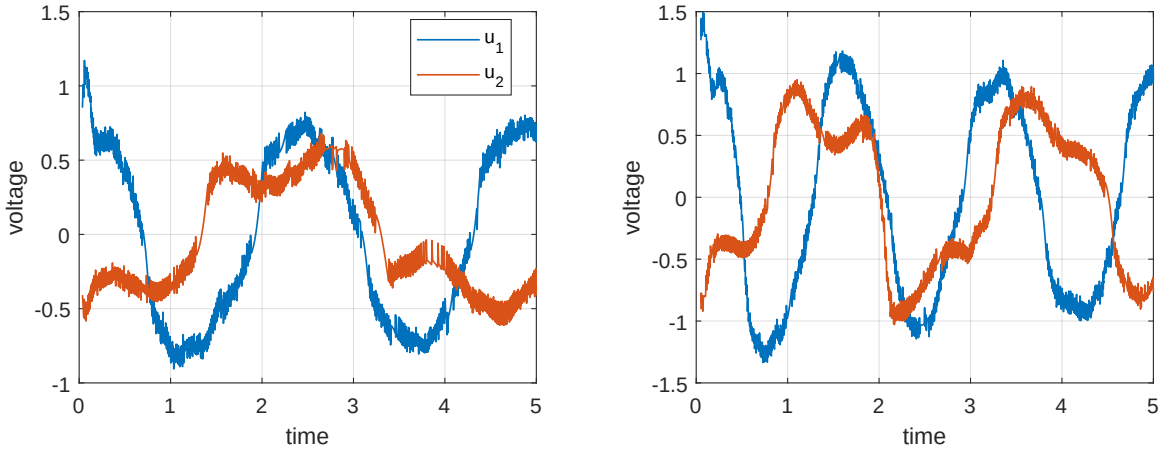
Additionally, we can compare the output voltages fed directly to the motors. Figure 7.15 presents these measurements for PD-only control architecture, whereas figures 7.16, 7.17 exhibit feedforward+PD architecture. Let us pinpoint that all figures presented in this section so far have been divided into parts (a) and (b), which allowed



(a) Trajectory $\mathcal{A}$ (b) Trajectory $\mathcal{B}$

Figure 7.14: Cartesian coordinates representation of the recreated trajectory compared with the reference for the feedforward/PD control architecture

us to conveniently convey the results of both experiments regarding trajectories $\mathcal{A}$ and $\mathcal{B}$, respectively. Conversely, figure 7.16 presents the results dedicated to trajectory $\mathcal{A}$ only. The left plot shows a feedback signal generated by the PD controller based on the current error. Concurrently, the right plot gathers feedforward signals (recorded apriori) with the measured output voltage supplied to the motors. The output voltage is a sum of the feedforward signal and the PD-generated corrections shown on the left plot. Figure 7.17 presents a similar set of plots corresponding to the trajectory $\mathcal{B}$. As might be expected, incorporating a feedforward signal greatly reduces error on the feedback controller, which is clearly visible upon a comparison between figure 7.15 and figs. 7.16, 7.17 (left plots).



(a) Trajectory $\mathcal{A}$ (b) Trajectory $\mathcal{B}$

Figure 7.15: Input motor voltages for PD-only control architecture

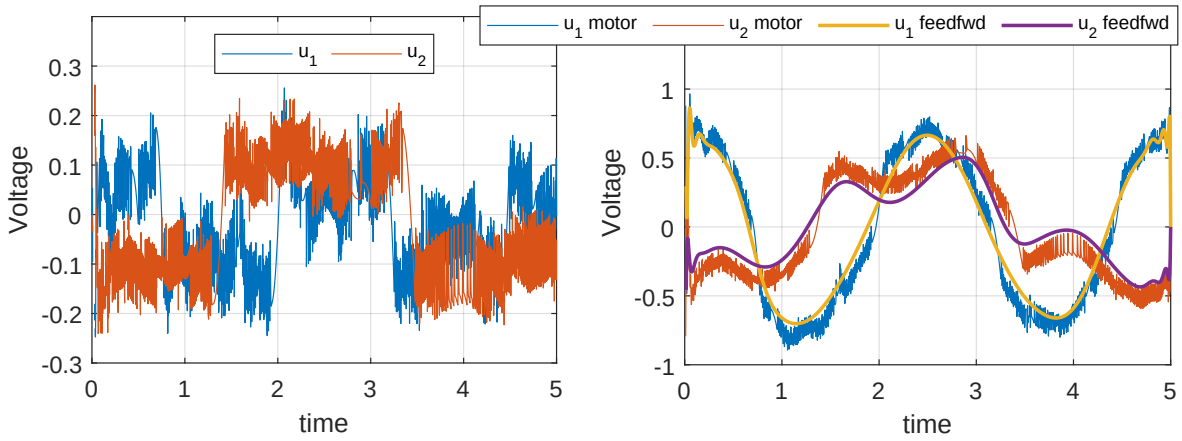


Figure 7.16: (left) PD controller voltage signal and (right) input motor voltages (rough lines) vs. feedforward voltage (solid lines); trajectory $\mathcal{A}$

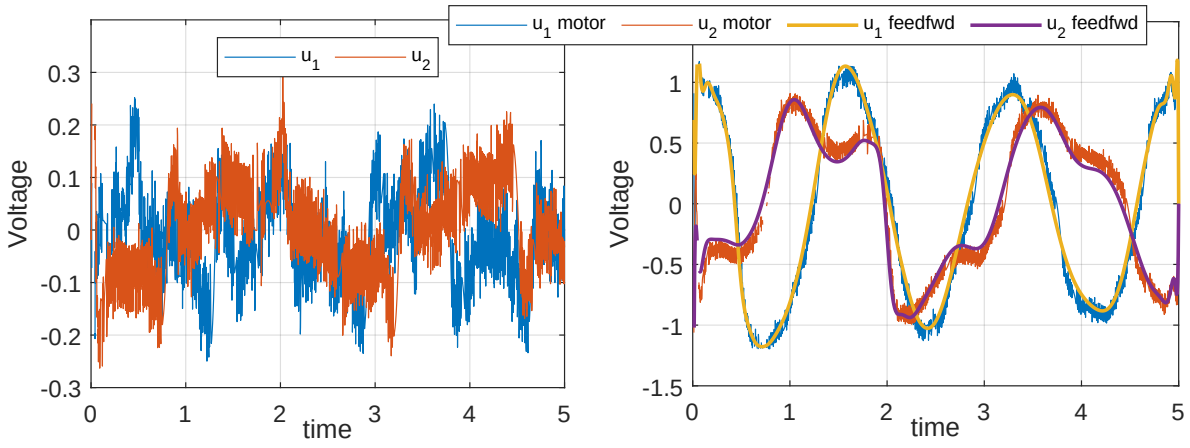


Figure 7.17: (left) PD controller voltage signal and (right) input motor voltages (rough lines) vs. feedforward voltage (solid lines); trajectory $\mathcal{B}$

7.6 Execution with constrained input signal

The approach developed throughout this work was successfully applied to solve the practical problem specified in section 7.3. It simultaneously provided internal reference variables for the PD controller and computed nominal feedforward signal. Nevertheless, the presented problem has been a rather typical task in robotics, obscuring some of the proposed method's benefits. To use the full potential of the adjoint-based optimal control approach, let us additionally limit the range of the control signal fed to the motors.

Generation of some trajectories may require applying prohibitively high voltage to the motors. In such cases, the trajectory can only partially be realized, resulting in high errors between reference and the outcome. As a remedy, it can be slightly altered not to exceed the maximum voltage. To this end, we define trajectory $\mathcal{C}$ based on eq. (7.1) ($\omega_x = 2\pi, \omega_y = \frac{4}{3}\omega_x, t_f = 3$ s) and optimize the cost functional (7.2) with additional

penalty term [75] for high magnitudes of input signal $\mathbf{u}(t) = [u_1(t), u_2(t)]^T$:

$$C(\mathbf{u}) = \gamma \int_0^{t_f} \big(P_1(u_1) + P_2(u_2)\big)\,\mathrm{d}t, \quad P_i(u_i) = \begin{cases} 0 & \text{for } |u_i| < u_i^{max} \\ \frac{1}{2}(|u_i| - u_i^{max})^2 & \text{otherwise,} \end{cases} \tag{7.4}$$

where $\gamma \in \mathcal{R}$ is a scaling factor.

Let us now assume that the maximum allowable voltage for the motors is $u_i^{max} = 2$ V, $i = \{1, 2\}$. We follow the same procedure, as in the previous section, i.e., first optimize eq. (7.2) with $\beta = \gamma = 0$, next we set $\beta = 1$, and ultimately $\beta = 1$, $\gamma = 0.2$. The initial guess is $u_i(t) = 0$, whereas the result of each procedure becomes the input approximation for the following optimization problem. This process has been summarized in figure 7.18.

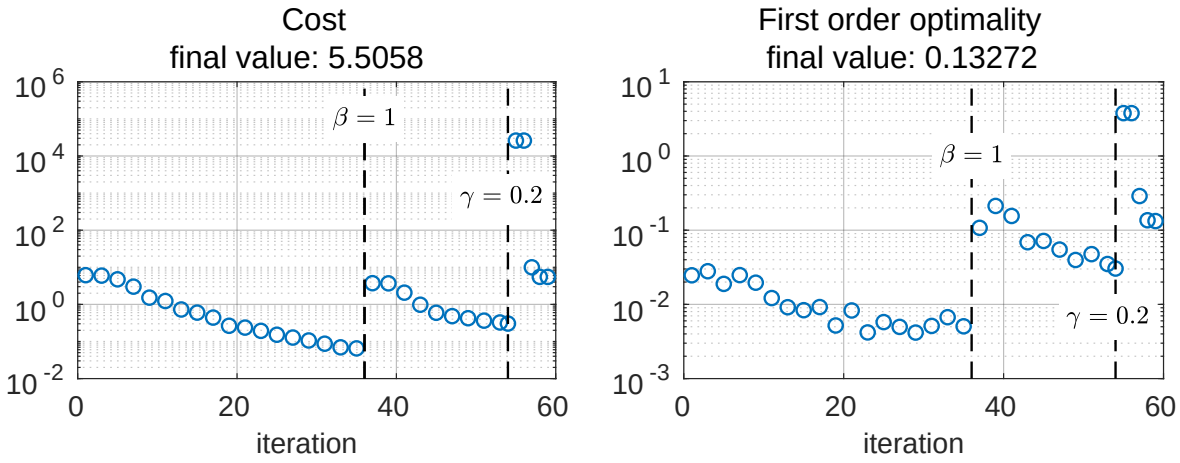
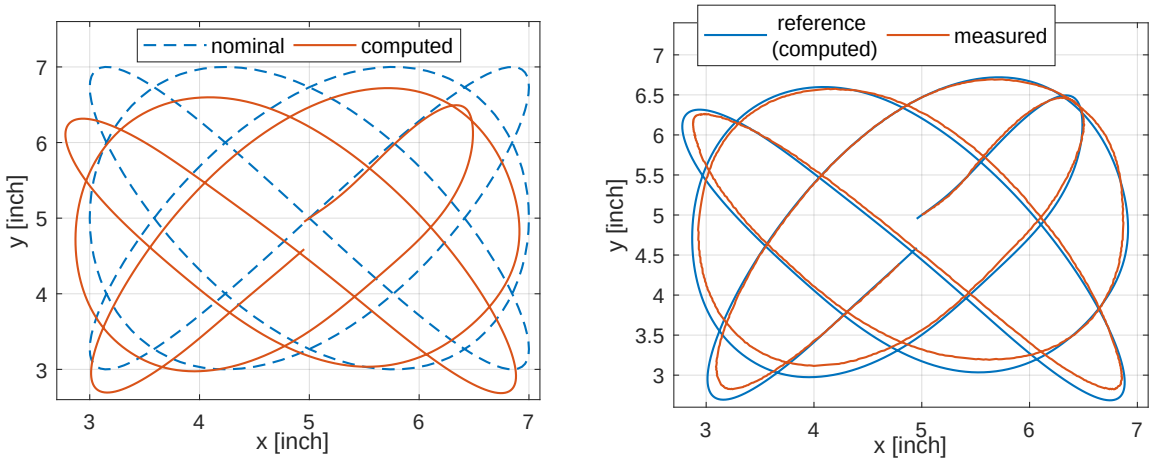


Figure 7.18: Optimization history for the model of a five-bar linkage following trajectory $\mathcal{C}$ with constrained input signal

Critically, the constraints imposed on the input signal prohibit the end-effector from following the trajectory defined in eq. 7.1. Therefore, The specified problem becomes significantly more complex, and classical inverse kinematics algorithms are no longer feasible to address this task. Nevertheless, the adjoint-based trajectory generator provided a new, slightly deformed reference path, ensuring appropriate bounds on the input voltage. The resultant (computed) trajectory has been compared with the nominal one in figure 7.19a, whereas the measured trajectory is depicted in figure 7.19b.

Additionally, figure 7.20 presents a time domain comparison between two executions: feedback-only and the other aided by a feedforward signal. Again, the direct comparison between internal variables is not convenient; hence figure 7.20b shows an error between the measured signal and the reference angles. Similarly to the previous cases, the errors are smaller when the feedforward signal is fed to the control loop.

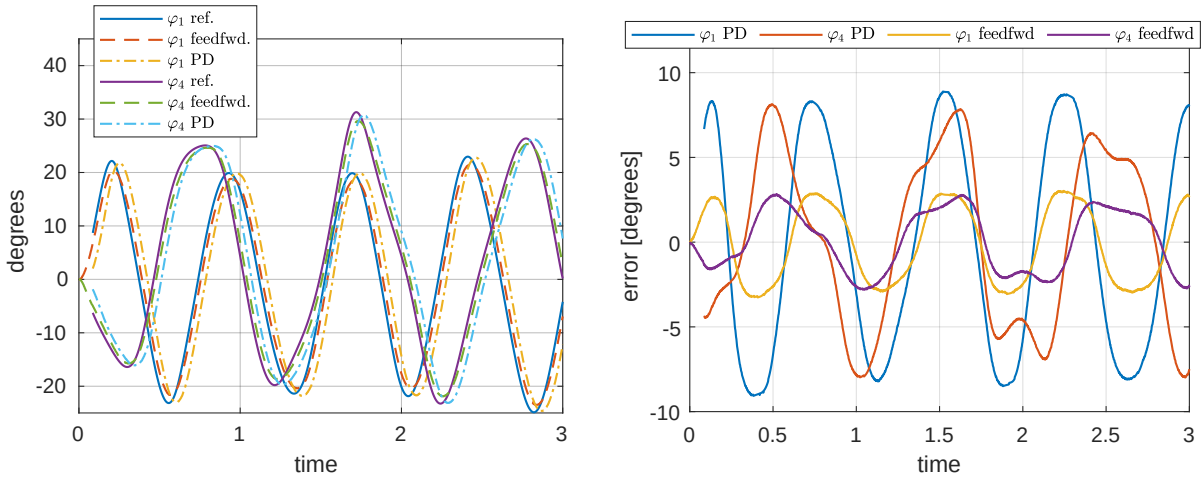


(a) optimization results

(b) hardware execution with feedforward

Figure 7.19: Cartesian coordinates representation of the trajectory $\mathcal{C}$: comparison between nominal, reference and generated trajectories

Last but not least, Figure 7.21 gathers multiple voltage signals generated during the execution on the hardware. The right plot shows the nominal (feedforward) signal fed to the control loop against the resultant voltage registered by the DAQ card. Simultaneously, the left plot depicts the corrections acquired from the PD controller that shape the resulting voltage fed to the motors.



(a) Measured angles

(b) Calculated error: $\varphi_i^{\text{ref}} - \varphi_i$

Figure 7.20: Joint angles measured for feedback-only and feedforward/feedback executions compared with their reference values

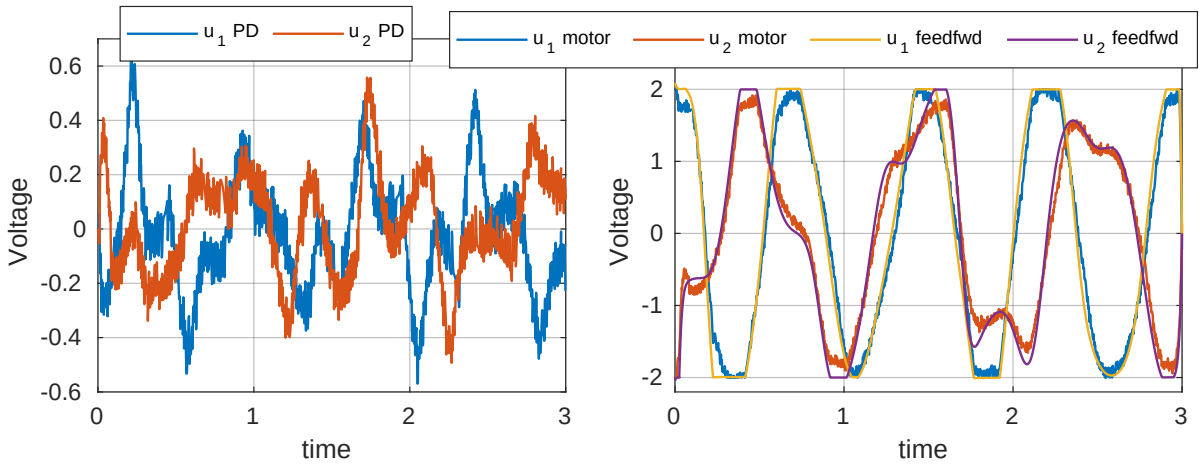


Figure 7.21: (left) PD controller voltage signal and (right) input motor voltages (rough lines) vs. feedforward voltage (solid lines); trajectory $\mathcal{C}$

7.7 Summary

Based on the five-bar linkage electromechanical model, we solved the optimal control problem in a simulation environment to generate a reference (feedforward) voltage signal and compute the reference joint variables. Subsequently, we fed the computed reference signals to the control loop during the execution on the actual device. The feedforward signal has been qualitatively correct, as indicated by figures 7.16, 7.17 (rightmost plots). However, due to the model uncertainty, accurate execution required a feedback action. Nevertheless, by applying a feedforward signal to the control loop, we were able to record more accurate control performance (c.f. fig. 7.12) combined with reduced error on the feedback controller, as substantiated by comparison between figure 7.15 and figs. 7.16, 7.17 (leftmost plots). Furthermore, the adjoint-based optimization algorithm was applied to solve a problem with constraints imposed on the input signal. The proposed procedure generated a feasible trajectory without violating prescribed bounds. Section 7.6 presented the outcome of the offline computations as well as the measurements as a part of real hardware experiment. Ultimately, it has been shown that the model-based analysis allows us to achieve better performance while simultaneously reducing the controller's burden during the online execution.

CHAPTER 8
CONCLUSIONS AND SUMMARY

The scope of this thesis integrates three main research fields, as visualized in fig. 2.2: multibody dynamics, optimal control theory, and parallel computing algorithms. The main goal involved the development of a computational procedure that solves complex large-scale mechanical problems via multibody dynamics analysis and utilizing the adjoint method. The novelty of the developed approach lies in the employment of the divide-and-conquer paradigm for the adjoint problem. The detailed derivation of the necessary background theory and the actual algorithm (chapters 3 and 4) were followed by parts presenting multiple examples: numerical test cases (ch. 5), software implementation with benchmarking of the HADCA (ch. 6), and experiments on a hardware (ch. 7).

All goals defined in section 2.5 have been fulfilled. It is worth emphasizing that each goal constitutes a scientific novelty in the state-of-the-art. Specifically, goals 1–3 contribute to adjoint sensitivity analysis by establishing a novel recursive approach based on the DCA. Goals 4 and 5 present applications in the simulation environment and hardware. The results obtained from these experiments comprise an additional novelty of this thesis.

Let us now address each goal individually.

Goal 1 Development of the Hamiltonian-based adjoint method in absolute coordinates – section 4.2

The adjoint system has been obtained via variational calculus of the performance measure. The formulation of the underlying EOM based on spatial velocities (3.14) is preferred to the corresponding formulation presented in section 3.2.1 and ref. [79] since it seamlessly integrates with the joint-space formulation. This property benefited from deriving the relationships between redundant and independent adjoint

variables in sec. 4.3, constituting a cornerstone for integrating the divide-and-conquer paradigm with the adjoint method.

The employed formulation of the underlying dynamics (based on the angular velocity) posed a challenge in the reliable computation of the partial derivatives for the coefficients of the adjoint system. To this end, appendix A pertains to specifying appropriate analytical derivatives with respect to the state variables in the spatial setting. The presented approach relies on defining variational relations between angular velocity and virtual rotation embodied by equation (A.7). A close agreement of the obtained results with the finite differences or complex step methods is a strong premise for the correctness of these computations.

Goal 2 Development of the Hamiltonian-based adjoint method in mixed absolute-joint coordinates – section 4.3

A typical formulation for EOM in multibody dynamics is a set of differential algebraic equations, e.g., eq. (3.14). These equations can be projected onto the motion subspace $\mathbf{H}$ to come up with unconstrained dynamic equations that describe the relations between joint-space variables. Consequently, by studying the direct relationships between joint-space and redundant formulations (c.f. eq. (4.24)), it was possible to encapsulate the relations between redundant and independent adjoint variables via eq. (4.26). Specified relations have been validated by conducting multiple numerical experiments.They allowed for the derivation of the adjoint system formulated as a set of ODEs (4.29). Ultimately, relationships between joint- and absolute-space variables were necessary to accomplish the goal.

Goal 3 Development of the recursive, divide-and-conquer adjoint method – section 4.5

This is one of the central goals of this thesis, and it heavily relies on the results achieved in previous research goals. Although applying the divide-and-conquer paradigm to the adjoint system has been a novelty in the state-of-the-art, specific properties of this approach remain similar to the methods dedicated to the forward dynamics problem. Precisely, fulfilling this goal required developing the following concepts that also appear in other DCA formulations:

- The adjoint of the dynamic system formulated redundantly, which allows for application of the divide-and-conquer paradigm (c.f. **Goal 1**)
- The adjoint of the dynamic system based on minimal set of coordinates and the relations between joint-space and redundant adjoint variables (c.f. **Goal 2**)

- The relations between absolute adjoint coordinates along a rigid body: equation (4.22)
- The relations between adjoint variables across arbitrary bodies, (the analog of relations between spatial velocities): equation (4.26)

The developed method has been thoroughly explained in the form of pseudocode describing each step of the algorithm in detail. Furthermore, a C++ code base has been written based on the specified algorithm and presented in chapter 6. This test case example has been used to measure the execution times of the algorithm, where the process variables were the size of the problem and the number of available computing threads. Based on the measurements, it was possible to evaluate the speedup of the developed method. Additionally, we analyzed and discussed multiple logical sections of the algorithm based on the measurements gathered in figure 6.8.

Goal 4 Implementation and testing of the developed methods in a simulation environment – chapter 5

The Hamiltonian-based adjoint method has been implemented as a Matlab computer code and tested against diversified examples, including optimal design and optimal control problems. All considered problems have been successfully solved via proposed algorithms. The outcome of specific examples was cross-examined with other methods, such as the servo-constraints method (c.f. figures 5.17 and 5.32).

Furthermore, the gradient obtained via the adjoint method was compared with different available approaches. For example, table 5.2 presents a detailed comparison of the results between the adjoint, direct differentiation, finite differences, and complex methods. Consequently, figures 5.15, 5.20, 5.28, and 5.33 depict the comparison between the proposed method and finite differences or complex-step method in the optimal control setting.

The validity of the developed relations between absolute and independent adjoint variables has been tested. The comparison was carried out in different spatial scenarios and has been depicted in figures 5.10, 5.15, and 5.20. Examples presented in chapter 5 were solved by means of redundant (global) formulation (4.14) since the number of bodies in the considered systems was low. The recursive formulation has been tested and benchmarked on a large-scale MBS in chapter 6.

Goal 5 Implementation of the developed methods in the hardware – chapter 7

The developed methods can be applied as feedforward control architecture, where the adjoint method yields a nominal (reference) input signal and the appropriate trajectory. Applying the adjoint method required constructing a mathematical model of the device presented in figure 7.2. The dynamic response of the model has been tested against the measured hardware response (c.f. fig. 7.5).

Section 7.5 has been devoted to discussing the investigation results, where feedforward-feedback control has been compared with a classical approach based on a PD controller. The results show that the proposed method significantly reduced the error signal fed to the PD controller and generated more accurate trajectories. Furthermore, the adjoint-based optimization algorithm was applied to solve a problem with constraints imposed on the input signal in section 7.6. The proposed procedure generated a feasible trajectory without violating prescribed bounds. These results were successively employed as a part of an actual hardware experiment.

The methods proposed in this thesis extend the state-of-the-art in optimal control and multibody dynamics theory. They can be applied in many practical applications in various research fields, such as robotics, machine design, or control. Upon application, the developed methods have a potential to improve the efficiency of computations, allowing for shifting the software-imposed limits closer to those imposed by the hardware.

The presented work has yet to exhaust the research algorithms' full potential and is expected to expand the derived approach in multiple directions. The HADCA algorithm can be generalized to involve closed-loop kinematic topology. This feature is currently under development by the author of the thesis. Subsequently, methods developed in the thesis might be implemented as a general purpose software suite allowing for simulation and adjoint sensitivity analysis of complex multibody systems. The software will implement all algorithms described herein and can be extended to include complex phenomena, such as discontinuities, contact, joints' flexibility, etc. Last but not least, we plan to deploy the optimal control algorithms on a KUKA LWR Robot [125] by integrating the adjoint method with, e.g., FPGA computation hardware [121]. It is expected to repeat the steps presented in chapter 7 with a more mature, industrial object to control.

APPENDIX A

VIRTUAL ROTATIONS

The calculus employed in section 4 heavily relies on variational principles. To simplify variations of certain quantities presented in section 3.2, we need to come up with a methodology of computing partial derivatives associated with the rotation matrix $\mathbf{A}$. To this end, we can utilize virtual rotations which possess handy vector properties. These quantities have been derived and thoroughly described in ref. [55]; however, some relations must be expanded to be utilized in the variational calculus presented in section 4.

The tilde operator $\sim$ provides a simple way to express vector product in terms of matrix operations. It is defined in equation (3.19) of section 3.3.1. The variation of the orientation matrix $\mathbf{A}$ can be expanded as $\delta\mathbf{A} = \mathbf{A}\delta\widetilde{\boldsymbol{\pi}}'$, where $\delta\boldsymbol{\pi}'$ is the virtual rotation of coordinate frame $\mathbf{A}$ with respect to the origin frame [55, 56]. Similarly, we can calculate the variation of an arbitrary vector $\mathbf{x}(\mathbf{q}) \in \mathcal{R}^3$ expressed in global reference frame as:

$$\delta\mathbf{x} = \delta(\mathbf{A}\mathbf{x}') = \mathbf{A}\delta\widetilde{\boldsymbol{\pi}}'\mathbf{x}' = -\mathbf{A}\widetilde{\mathbf{x}}'\delta\boldsymbol{\pi}' = \mathbf{x}_{\boldsymbol{\pi}'}^{\text{expl}} \cdot \delta\boldsymbol{\pi}'. \tag{A.1}$$

An apostrophe behind the symbol indicates that the quantity is expressed in the local reference frame and treated as constant (hence $\delta\mathbf{x}' = \mathbf{0}$). Furthermore, the quantity $\mathbf{x}_{\boldsymbol{\pi}'}^{\text{expl}}$ can be treated as an *explicit* partial derivative of $\mathbf{x}$ with respect to the orientation.

The adjoint method presented in this work requires calculating the variation of an expression involving velocities. Consequently, we need to express the variation of angular velocity with respect to the virtual rotation and its derivative. Since the variational and differential operators commute (for the classes of functions investigated in this work), we can derive the following set of equations:

$$\dot{\mathbf{A}} = \mathbf{A}\widetilde{\boldsymbol{\omega}'}, \tag{A.2a}$$

$$\delta\,\dot{\mathbf{A}} = \delta\mathbf{A}\widetilde{\boldsymbol{\omega}'} + \mathbf{A}\delta\widetilde{\boldsymbol{\omega}'}, \tag{A.2b}$$

$$\delta\dot{\mathbf{A}} = \mathbf{A}\delta\widetilde{\boldsymbol{\pi}'}\widetilde{\boldsymbol{\omega}'} + \mathbf{A}\delta\widetilde{\boldsymbol{\omega}'}, \tag{A.2c}$$

$$\delta\mathbf{A} = \mathbf{A}\delta\widetilde{\boldsymbol{\pi}'}, \tag{A.3a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\,\delta\mathbf{A} = \dot{\mathbf{A}}\delta\widetilde{\boldsymbol{\pi}'} + \mathbf{A}\delta\dot{\widetilde{\boldsymbol{\pi}'}}, \tag{A.3b}$$

$$\delta\dot{\mathbf{A}} = \mathbf{A}\widetilde{\boldsymbol{\omega}'}\delta\widetilde{\boldsymbol{\pi}'} + \mathbf{A}\delta\dot{\widetilde{\boldsymbol{\pi}'}}. \tag{A.3c}$$

Let us premultiply equations (A.2c) and (A.3c) by $\mathbf{A}^T$. Since their left-hand side is equal, we can compare their right hand sides:

$$\begin{aligned}
\delta\widetilde{\boldsymbol{\pi}'}\widetilde{\boldsymbol{\omega}'} + \delta\widetilde{\boldsymbol{\omega}'} &= \widetilde{\boldsymbol{\omega}'}\delta\widetilde{\boldsymbol{\pi}'} + \delta\dot{\widetilde{\boldsymbol{\pi}'}}, \\
\delta\widetilde{\boldsymbol{\omega}'} &= \delta\dot{\widetilde{\boldsymbol{\pi}'}} + \widetilde{\boldsymbol{\omega}'}\delta\widetilde{\boldsymbol{\pi}'} - \delta\widetilde{\boldsymbol{\pi}'}\widetilde{\boldsymbol{\omega}'}.
\end{aligned} \tag{A.4}$$

By invoking the identity $\widetilde{(\widetilde{\mathbf{a}}\mathbf{b})} = \widetilde{\mathbf{a}}\widetilde{\mathbf{b}} - \widetilde{\mathbf{b}}\widetilde{\mathbf{a}}$ [56] one can write eq. (A.4) as:

$$\begin{aligned}
\delta\widetilde{\boldsymbol{\omega}'} &= \delta\dot{\widetilde{\boldsymbol{\pi}'}} + \widetilde{\widetilde{\boldsymbol{\omega}'}\delta\boldsymbol{\pi}'} \\
\delta\boldsymbol{\omega}' &= \delta\dot{\boldsymbol{\pi}'} + \widetilde{\boldsymbol{\omega}'}\delta\boldsymbol{\pi}'.
\end{aligned} \tag{A.5}$$

Let us now consider an arbitrary expression dependent on the generalized coordinates and absolute velocity vectors, i.e.: $\mathbf{x}(\mathbf{q},\mathbf{v}) = \mathbf{x}(\mathbf{r},\mathbf{e},\dot{\mathbf{r}},\boldsymbol{\omega}')$. Equation (A.5) allows calculating its total variation as:

$$\begin{aligned}
\delta\mathbf{x} &= \mathbf{x}_{\mathbf{r}}\delta\mathbf{r} + \mathbf{x}_{\dot{\mathbf{r}}}\delta\dot{\mathbf{r}} + \mathbf{x}^{\mathrm{expl}}_{\boldsymbol{\pi}'}\delta\boldsymbol{\pi}' + \mathbf{x}_{\boldsymbol{\omega}'}\delta\boldsymbol{\omega}' \\
&= \mathbf{x}_{\mathbf{r}}\delta\mathbf{r} + \mathbf{x}_{\dot{\mathbf{r}}}\delta\dot{\mathbf{r}} + \left(\mathbf{x}^{\mathrm{expl}}_{\boldsymbol{\pi}'} + \mathbf{x}_{\boldsymbol{\omega}'}\widetilde{\boldsymbol{\omega}'}\right)\delta\boldsymbol{\pi}' + \mathbf{x}_{\boldsymbol{\omega}'}\delta\dot{\boldsymbol{\pi}}'.
\end{aligned} \tag{A.6}$$

According to equation (A.6), the total variation of $\mathbf{x}$ consists of two independent variations: virtual rotation (displacement) and its derivative. Moreover, eq. (A.5) suggests that the term combined with the angular velocity involves an additional, implicit quantity that must be considered when computing partial differences associated with virtual rotation $\delta\boldsymbol{\pi}'$. The expression in parentheses in eq. (A.6) is thus the complete partial difference $\mathbf{x}_{\boldsymbol{\pi}'}$ with respect to the virtual rotation.

The notation above aims to pinpoint the low–level details of the calculations; however, it will be more convenient to write the derivatives with respect to the stacked vector of generalized coordinates when deriving the adjoint system. State variations in stacked vector format are described by eq. (4.5), whereas eqs. (A.5) and (A.6) have the following form in global formulation, respectively:

$$\delta\mathbf{v} = \delta\dot{\mathbf{s}} + \boldsymbol{\Omega}\;\delta\mathbf{s}, \tag{A.7}$$

$$\delta\mathbf{x} = \mathbf{x}_\mathbf{s}^{\text{expl}}\ \delta\mathbf{s} + \mathbf{x}_\mathbf{v}\delta\mathbf{v} \tag{A.8a}$$

$$\begin{aligned} &= \left(\mathbf{x}_\mathbf{s}^{\text{expl}} + \mathbf{x}_\mathbf{v}\boldsymbol{\Omega}\right)\ \delta\mathbf{s} + \mathbf{x}_\mathbf{v}\ \delta\dot{\mathbf{s}} \\ &= \mathbf{x}_\mathbf{s}\ \delta\mathbf{s} + \mathbf{x}_\mathbf{v}\delta\dot{\mathbf{s}}, \end{aligned} \tag{A.8b}$$

where

$$\boldsymbol{\Omega} = 2\mathbf{L}_d\dot{\mathbf{L}}_h^T = \begin{bmatrix} \ddots & & & \\ & \mathbf{0}_{3\times3} & \mathbf{0} & \\ & \mathbf{0} & \widetilde{\boldsymbol{\omega}}_i & \\ & & & \ddots \end{bmatrix} = \mathbf{v}_\mathbf{s}, \tag{A.9}$$

and $\mathbf{L}_d$, $\mathbf{L}_h$ are defined in eq. (3.12). Equations (A.8) convey important information that variation of an arbitrary expression can be calculated in two ways. The implicit term involving the product $\boldsymbol{\Omega}\ \delta\mathbf{s}$ can be either a part of the variation $\delta\mathbf{v}$ (as in eq. (A.8a)), or it can be explicitly added to the term $\mathbf{x}_\mathbf{s}^{\text{expl}}$ (as in eq. (A.8b)), yielding a complete partial derivative $\mathbf{x}_\mathbf{s}$. In this work, we implement the latter approach, where partial derivatives with respect to the operator $\mathbf{s}$ (or more accurately $\boldsymbol{\pi}'$) take into account the implicit component $\boldsymbol{\Omega}\ \delta\mathbf{s}$. Example A.1 illustrates how this convention is applied in practical calculations.

Example A.1. Calculate the partial derivative of the kinetic energy T with respect to the virtual rotation, assuming that all coordinate frames are attached to the bodies' centers of mass.

The kinetic energy reads $T = \frac{1}{2}\mathbf{v}^T\mathbf{M}\mathbf{v}$. Under the last assumption, we can describe the mass matrix as constant and block–diagonal (note that inertia is expressed in the body reference frame). Therefore, only rotational component, i.e. $\frac{1}{2}\boldsymbol{\omega}'^T\mathbf{J}'\boldsymbol{\omega}'$ will produce a non–zero result. We can thus write the total variation of T as:

$$\delta T = \delta\big(\frac{1}{2}\boldsymbol{\omega}'^T\mathbf{J}'\boldsymbol{\omega}'\big) = \boldsymbol{\omega}'^T\mathbf{J}'\ \delta\boldsymbol{\omega}' \overset{(A.5)}{=} \boldsymbol{\omega}'^T\mathbf{J}'\ \delta\dot{\boldsymbol{\pi}} + \underbrace{\boldsymbol{\omega}'^T\mathbf{J}'\widetilde{\boldsymbol{\omega}'}}_{T_{\boldsymbol{\pi}'}}\ \delta\boldsymbol{\pi}' \tag{A.10}$$

The partial differential quantity of T is the coefficient standing next to the virtual rotation $\delta\boldsymbol{\pi}'$, i.e., $T_{\boldsymbol{\pi}'}^T = \widetilde{\boldsymbol{\omega}'}^T\mathbf{J}'\boldsymbol{\omega}'$. Let us pinpoint that, similarly to the Jacobian $\mathbf{C}$, $T_{\boldsymbol{\pi}'}$ is not a partial derivative, rather a Jacobian matrix equivalent to $T_\mathbf{e} = 2T_{\boldsymbol{\pi}'}\mathbf{L}$.

. . .

Let us now shift the focus to the necessary conditions required for the extremum of a functional. As explained in chapter 4, the key idea of the adjoint method is to bind

the variation of the performance measure solely to the variation of the vector of design variables. The adjoint system rises directly from equating to zero coefficients appearing next to variations of other variables in eq. (4.7). To come up with equation (4.7), we utilized a simplified form of eq. (A.7). In the following part, we will explain why it is necessary to omit the term $\boldsymbol{\Omega}\ \delta\mathbf{s}$ when deriving the adjoint system in the specified formulation.

To this end, we define a Lagrange multiplier $\boldsymbol{\varphi}(t) \in \mathcal{R}^{n_\beta}$ and an arbitrary function $\mathbf{x}(\mathbf{q}, \mathbf{v}, \mathbf{b}) \in \mathcal{R}^{n_\beta}$. Next, we apply the same calculations as those performed in section 4.2 over the extended performance measure, with a sole distinction that currently we employ the general form of eq. (A.7):

$$
\begin{aligned}
\delta \int_0^{t_f} -\boldsymbol{\varphi}^T\mathbf{x}\,\mathrm{d}t &= \int_0^{t_f} \Big[-\boldsymbol{\varphi}^T\mathbf{x}_\mathbf{v}\delta\mathbf{v} - \boldsymbol{\varphi}^T\mathbf{x}_\mathbf{s}\,\delta\mathbf{s} - \boldsymbol{\varphi}^T\mathbf{x}_\mathbf{b}\delta\mathbf{b}\Big]\,\mathrm{d}t \\
&\overset{(A.7)}{=} \int_0^{t_f} \Big[-\boldsymbol{\varphi}^T\mathbf{x}_\mathbf{v}\,\delta\dot{\mathbf{s}} - \boldsymbol{\varphi}^T\mathbf{x}_\mathbf{s}\,\delta\mathbf{s} - \cancel{\boldsymbol{\varphi}^T\mathbf{x}_\mathbf{v}\boldsymbol{\Omega}\,\delta\mathbf{s}} - \boldsymbol{\varphi}^T\mathbf{x}_\mathbf{b}\delta\mathbf{b}\Big]\,\mathrm{d}t \quad \rightarrow \text{(by parts)} \\
&= \int_0^{t_f} \Big[\frac{\mathrm{d}}{\mathrm{d}t}\Big(\boldsymbol{\varphi}^T\mathbf{x}_\mathbf{v}\Big)\,\delta\mathbf{s} - \boldsymbol{\varphi}^T\mathbf{x}_\mathbf{s}\,\delta\mathbf{s} - \cancel{\boldsymbol{\varphi}^T\mathbf{x}_\mathbf{v}\boldsymbol{\Omega}\,\delta\mathbf{s}} - \boldsymbol{\varphi}^T\mathbf{x}_\mathbf{b}\delta\mathbf{b}\Big]\,\mathrm{d}t - \boldsymbol{\varphi}^T\mathbf{x}_\mathbf{v}\,\delta\mathbf{s}\Big|_{t_f} \\
&= \int_0^{t_f} \Big\{\Big[\dot{\boldsymbol{\varphi}}^T\mathbf{x}_\mathbf{v} - \boldsymbol{\varphi}^T\Big(\mathbf{x}_\mathbf{s} + \cancel{\boldsymbol{\varphi}^T\mathbf{x}_\mathbf{v}\boldsymbol{\Omega}} - \frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{x})_\mathbf{v}\Big)\Big]\,\delta\mathbf{s} - \boldsymbol{\varphi}^T\mathbf{x}_\mathbf{b}\delta\mathbf{b}\Big\}\,\mathrm{d}t - \boldsymbol{\varphi}^T\mathbf{x}_\mathbf{v}\,\delta\mathbf{s}\Big|_{t_f}.
\end{aligned}
\tag{A.11}
$$

The task is to specify necessary conditions arising from the variation $\delta\mathbf{s}$ by gathering coefficients standing next to it and equating the whole expression to zero. While eq. (A.11) describes a variational relation, the resultant differential (algebraic) equations (4.14) become separated from the variational context (i.e., the variation $\delta\mathbf{s}$ is omitted).

Now, let us investigate the geometrical properties of the quantities $\boldsymbol{\omega}'$ and $\delta\boldsymbol{\pi}'$ that constitute the product $\boldsymbol{\Omega}\ \delta\mathbf{s}$. Both of these are vector quantities describing rotational properties. In fact, equations (A.2a) and (A.3a) suggest, that these vectors are parallel to each other. Consequently, the expression $\widetilde{\boldsymbol{\omega}'}\delta\boldsymbol{\pi}'$ signifies a vector product, which is equal to zero for parallel vectors. This geometrical property must be considered before articulating necessary conditions for the extremum of δJ since it affects the resultant shape of the adjoint system. Hence, equation (A.7) must be reduced to a simple form $\delta\mathbf{v} = \delta\dot{\mathbf{s}}$ when deriving the adjoint system. Critically, the presented approach was successfully used in multiple problems involving spatial MBS, e.g., the linkage presented in section 5.5.

Example A.2 provides the reader with a rudimentary knowledge of how the calculations derived in this section can be applied practically over the performance measure. The example presents only a single simple component (that can be, e.g., associated with the velocity constraints function) dependent on the position/orientation and velocity to pinpoint the main issues that must be taken into account while deriving the adjoint system.

Example A.2. Let us consider a functional $I = \int_0^{t_f} \boldsymbol{\mu}^T \dot{\boldsymbol{\Phi}} \, \mathrm{d}t$ that consists of an arbitrary Lagrange multiplayer $\boldsymbol{\mu} \in \mathcal{R}^m$ and the derivative of geometric constraints vector, i.e. $\dot{\boldsymbol{\Phi}}(\mathbf{q}, \mathbf{v}, \mathbf{b}) \in \mathcal{R}^{m \times n}$. Calculate the variation of the functional and integrate higher-order term by parts to relate I with $\delta\mathbf{s}$ and $\delta\mathbf{b}$.

Let us recall that $\dot{\boldsymbol{\Phi}} = \mathbf{C}\mathbf{v}$. The variation of the functional can be written as:

$$\delta I = \int_0^{t_f} \left[\boldsymbol{\mu}^T \mathbf{C} \delta\mathbf{v} + \boldsymbol{\mu}^T (\mathbf{C}\mathbf{v})_{\mathrm{s}} \, \delta\mathbf{s} \right] \mathrm{d}t. \tag{A.12}$$

The substitution $\delta\mathbf{v} = \delta\dot{\mathbf{s}}$ and integration by parts yields:

$$\begin{aligned} \delta I &= \int_0^{t_f} \left[-\dot{\boldsymbol{\mu}}^T \mathbf{C} \, \delta\mathbf{s} - \boldsymbol{\mu}^T \dot{\mathbf{C}} \, \delta\mathbf{s} + \boldsymbol{\mu}^T (\mathbf{C}\mathbf{v})_{\mathrm{s}} \, \delta\mathbf{s}. \right] \mathrm{d}t + \boldsymbol{\mu}^T \mathbf{C} \, \delta\mathbf{s}\big|_{t_f} \\ &= \int_0^{t_f} \left[-\dot{\boldsymbol{\mu}}^T \mathbf{C} \, \delta\mathbf{s} + \boldsymbol{\mu}^T \left((\dot{\boldsymbol{\Phi}})_{\mathrm{s}} - \dot{\mathbf{C}} \right) \delta\mathbf{s} \right] \mathrm{d}t + \boldsymbol{\mu}^T \mathbf{C} \, \delta\mathbf{s}\big|_{t_f} . \end{aligned} \tag{A.13}$$

It can be shown that for many rudimentary constraint equations the following property exists: $(\dot{\boldsymbol{\Phi}})_{\mathrm{s}} = \dot{\mathbf{C}}$, which simplifies the total variation to the following form:

$$\delta I = \int_0^{t_f} -\dot{\boldsymbol{\mu}}^T \mathbf{C} \, \delta\mathbf{s} \, \mathrm{d}t + \boldsymbol{\mu}^T \mathbf{C} \, \delta\mathbf{s}\big|_{t_f} . \tag{A.14}$$

An interesting observation arises that the variable $\boldsymbol{\mu}$ does not appear under the integral of the performance measure. As a result, the right–hand side of the adjoint system does not explicitly depend on this variable.

APPENDIX B

RIGHT-HAND SIDE OF THE ADJOINT SYSTEM

Equation (4.14) defines the adjoint system to Hamilton's canonical dynamical equations of motion. This section is devoted to deriving the quantities on the right–hand side of the adjoint system. Their accurate representation is crucial to obtain the meaningful results of adjoint variables, and consequently, the gradient of the performance measure. First, let us rewrite equation (4.3) and mark some components.

$$\overline{J} = \int_0^{t_f} \Big[h + \boldsymbol{\eta}^T \big(\mathbf{p}^* - \underbrace{\mathbf{M}\mathbf{v}}_{(1)} - \underbrace{\mathbf{C}^T\boldsymbol{\sigma}}_{(2)} \big) + \boldsymbol{\xi}^T \big(\dot{\mathbf{p}}^* - \underbrace{\mathbf{f}}_{(3)} - \underbrace{\dot{\mathbf{C}}^T\boldsymbol{\sigma}}_{(4)} \big) - \underbrace{\boldsymbol{\mu}^T\dot{\boldsymbol{\Phi}}}_{(5)} \Big] \, \mathrm{d}t. \tag{B.1}$$

The unmarked components, such as $h, \mathbf{p}^*, \dot{\mathbf{p}}^*$, are simple enough to be derived directly in section 4.2 (c.f. eq. (4.10c)). Moreover, the term (5) was calculated in example A.2 and shown not to appear on the RHS of the adjoint system. The partial differences of the remaining components (1) – (4) appear on the RHS of the adjoint system, hence they constitute the vector $\mathbf{r}$ from eq. (4.14). The goal is then to derive a universal framework that clearly shows how these quantities transform from eq. (B.1) to the components of the vector $\mathbf{r}$. To this end, we define a Lagrange multiplier $\boldsymbol{\varphi}(t) \in \mathcal{R}^{n_\beta}$ and an arbitrary function $\mathbf{x}(\mathbf{q}, \mathbf{v}) \in \mathcal{R}^{n_\beta}$. Next, we apply the same calculations as those performed in section 4.2 over the extended performance measure:

$$\begin{aligned}
\delta \int_0^{t_f} -\boldsymbol{\varphi}^T\mathbf{x} \, \mathrm{d}t &= \int_0^{t_f} -\boldsymbol{\varphi}^T\mathbf{x}_\mathbf{v}\delta\mathbf{v} - \boldsymbol{\varphi}^T\mathbf{x}_\mathbf{s} \, \delta\mathbf{s} \, \mathrm{d}t \\
&= \int_0^{t_f} -\boldsymbol{\varphi}^T\mathbf{x}_\mathbf{v} \, \delta\dot{\mathbf{s}} - \boldsymbol{\varphi}^T\mathbf{x}_\mathbf{s} \, \delta\mathbf{s} \, \mathrm{d}t \qquad \rightarrow \text{(by parts)} \\
&= \int_0^{t_f} \frac{\mathrm{d}}{\mathrm{d}t}\Big(\boldsymbol{\varphi}^T\mathbf{x}_\mathbf{v}\Big) \, \delta\mathbf{s} - \boldsymbol{\varphi}^T\mathbf{x}_\mathbf{s} \, \delta\mathbf{s} \, \mathrm{d}t - \boldsymbol{\varphi}^T\mathbf{x}_\mathbf{v} \, \delta\mathbf{s}\big|_{t_f} \\
&= \int_0^{t_f} \Big[\dot{\boldsymbol{\varphi}}^T\mathbf{x}_\mathbf{v} - \boldsymbol{\varphi}^T\Big(\mathbf{x}_\mathbf{s} - \frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{x})_\mathbf{v}\Big)\Big] \, \delta\mathbf{s} \, \mathrm{d}t - \boldsymbol{\varphi}^T\mathbf{x}_\mathbf{v} \, \delta\mathbf{s}\big|_{t_f} .
\end{aligned} \tag{B.2}$$

As an example, let us consider the term denoted with (1) by substituting $\boldsymbol{\varphi} := \boldsymbol{\eta}$ and $\mathbf{x} := \mathbf{M}\mathbf{v}$. By following the pattern presented in eq. (B.2), we come up with:

$$\delta \int_0^{t_f} -\boldsymbol{\eta}^T \mathbf{M}\mathbf{v}\, \mathrm{d}t = \int_0^{t_f} \left[\dot{\boldsymbol{\eta}}^T \mathbf{M} - \boldsymbol{\eta}^T \Big((\mathbf{M}\mathbf{v})_{\mathrm{s}} - \dot{\mathbf{M}} \Big)\right] \delta\mathbf{s}\, \mathrm{d}t - \boldsymbol{\eta}^T \mathbf{M}\, \delta\mathbf{s}\big|_{t_f}. \tag{B.3}$$

A closer look at eq. (B.2) reveals the structure of the adjoint system (4.11b). The term $(\dot{\boldsymbol{\varphi}}^T \mathbf{x}_{\mathrm{v}})^T$ stands on the left-hand side of the equation, whereas the terminal quantity $(\boldsymbol{\varphi}^T \mathbf{x}_{\mathrm{v}})^T\big|_{t_f}$ appears on the left-hand side of the boundary conditions (4.13c). Finally, the term $-\boldsymbol{\varphi}^T \Big(\mathbf{x}_{\mathrm{s}} - \frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{x})_{\mathrm{v}} \Big)$ resembles the RHS of the quantity $\mathbf{r}_{\mathcal{A}}$ in eq. (4.8).

Equation (4.10) was transformed into the equivalent form (4.11), which influenced the right–hand side of the resultant adjoint system. Ultimately, the RHS term can be written as:

$$\begin{aligned} \mathbf{r} &= \left(h_{\mathrm{s}} - \frac{\mathrm{d}}{\mathrm{d}t} h_{\mathrm{v}}\right)^T - \left(\overset{\sim}{\mathbf{p}}{}^*_{\mathrm{s}} - \dot{\mathbf{M}} - \overset{\sim}{\dot{\mathbf{p}}}{}^*_{\mathrm{v}}\right)^T \boldsymbol{\eta} - \left(\overset{\sim}{\dot{\mathbf{p}}}{}^*_{\mathrm{s}} - \frac{\mathrm{d}}{\mathrm{d}t} \overset{\sim}{\dot{\mathbf{p}}}{}^*_{\mathrm{v}}\right)^T \boldsymbol{\xi} \\ &= \left(h_{\mathrm{s}} - \frac{\mathrm{d}}{\mathrm{d}t} h_{\mathrm{v}}\right)^T - \Big((\mathbf{M}\mathbf{v})_{\mathrm{s}} + (\mathbf{C}^T\boldsymbol{\sigma})_{\mathrm{s}} - \dot{\mathbf{M}} - \mathbf{f}_{\mathrm{v}} - (\dot{\mathbf{C}}^T\boldsymbol{\sigma})_{\mathrm{v}} \Big)^T \boldsymbol{\eta} \\ &\qquad - \left(\mathbf{f}_{\mathrm{s}} + (\dot{\mathbf{C}}^T\boldsymbol{\sigma})_{\mathrm{s}} - \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{f}_{\mathrm{v}} - \frac{\mathrm{d}}{\mathrm{d}t}(\dot{\mathbf{C}}^T\boldsymbol{\sigma})_{\mathrm{v}} \right)^T \boldsymbol{\xi}. \end{aligned} \tag{B.4}$$

We can simplify eq. (B.4) by taking into account certain relationships. First of all, it can be shown that $\dot{\mathbf{C}} = (\dot{\boldsymbol{\Phi}})_{\mathrm{s}} = (\mathbf{C}\mathbf{v})_{\mathrm{s}}$. Since the variations $\delta\mathbf{s}, \delta\mathbf{v}$ are independent of the variable $\boldsymbol{\sigma}$, a similar property holds:

$$\left(\dot{\mathbf{C}}^T\boldsymbol{\sigma}\right)_{\mathrm{v}} = \left((\mathbf{C}\mathbf{v})_{\mathrm{s}}^T\boldsymbol{\sigma}\right)_{\mathrm{v}} = \left((\mathbf{C}\mathbf{v})_{\mathrm{v}}^T\boldsymbol{\sigma}\right)_{\mathrm{s}} = (\mathbf{C}^T\boldsymbol{\sigma})_{\mathrm{s}}. \tag{B.5}$$

Moreover, by employing the property $\dot{\boldsymbol{\sigma}} = \boldsymbol{\lambda}$, the last term in eq. (B.4) can be expanded as:

$$\frac{\mathrm{d}}{\mathrm{d}t}(\dot{\mathbf{C}}^T\boldsymbol{\sigma})_{\mathrm{v}} = \frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{C}^T\boldsymbol{\sigma})_{\mathrm{s}} = (\dot{\mathbf{C}}^T\boldsymbol{\sigma})_{\mathrm{s}} + (\mathbf{C}^T\boldsymbol{\lambda})_{\mathrm{s}} \tag{B.6}$$

Inserting both relations into eq. (B.4) simplifies it into the following form:

$$\mathbf{r} = \left(h_{\mathrm{s}} - \frac{\mathrm{d}}{\mathrm{d}t} h_{\mathrm{v}}\right)^T - \Big((\mathbf{M}\mathbf{v})_{\mathrm{s}} - \dot{\mathbf{M}} - \mathbf{f}_{\mathrm{v}} \Big)^T \boldsymbol{\eta} - \left(\mathbf{f}_{\mathrm{s}} - \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{f}_{\mathrm{v}} - (\mathbf{C}^T\boldsymbol{\lambda})_{\mathrm{s}} \right)^T \boldsymbol{\xi}. \tag{B.7}$$

APPENDIX C

SUPPLEMENTARY ALGORITHMS

Chapter 6 presented a detailed algorithm implementing the Hamiltonian Adjoint-based Divide and Conquer Algorithm. The computation of gradient at a given step of the optimization procedure requires one to perform recursions over the following quantities:

1. velocity equations (3.14a) (figure 3.5, algorithm 2) ,
2. force distribution (3.47) (figure 3.6, algorithm 4) ,
3. acceleration equations (3.15) (algorithm 5) ,
4. adjoint system (4.14b) (figure 4.2, algorithm 3) .

Algorithm 2 is too extensive to be presented as a whole; hence it has been divided into three logical units: the central part yielding joint velocities of a MBS, algorithm 4 evaluating articulated forces of each subassembly, and alg. 5, that computes joint accelerations required for calculating appropriate derivatives used by the adjoint method. Let us note that alg. 5 does not have to be run as many times as algorithms 2 and 4 since its purpose is to evaluate accelerations at user-defined time steps (e.g., at uniformly distributed grid). On the other hand, algorithms 2 and 4 are called by a variable-step integration routine and must be executed at the intermediate steps to meet the assigned error tolerances.

Algorithm 4 Evaluation of articulated forces (HDCA)

Inputs: $\boldsymbol{\alpha}^k, \dot{\boldsymbol{\alpha}}^k, u^k$, `Tree`
Global variables: `Nbodies, Nthreads, Ntiers, input`

Numerical procedure:
Leaf-body level calculations

```
#omp for(i=0; i < Nbodies; ++i)                          ▷ TforceLeaves: Tic
  Tree[0] ← computeForceAtHandle1(α^k, α̇^k, u^k, input) (fig. 3.6)
end #omp for                                             ▷ TforceLeaves: Toc
```

Recursion from leaves to root node

```
for (i=1; i < Ntiers; ++i)                               ▷ TforceAsm: Tic
  upBranch ← Tree[i-1]  /* upperBranch */
  assembliesInBranch ← readNoAssemblies(i, Nbodies)
  #omp for(j=0; j < assembliesInBranch; ++j)
    Tree[i] ← accumulateForces(upBranch[2j], upBranch[2j+1]) (3.49)
  end #omp for
  if (assembliesInBranch is odd)
    Tree[i][end] ← upperBranch[end]
end for                                                  ▷ TforceAsm: Toc
```

Backward recursion from root node S to leaf bodies

```
Q̄_1^S ← Q_1^S, Q̄_2^S ← 0
for (i=Ntiers-2; i > 0; --i)                             ▷ TforceDis: Tic
  assembliesInBranch ← readNoAssemblies(i, Nbodies)
  #omp for(j=0; j < assembliesInBranch; ++j)
    Tree[i][j].disassemble() (3.50)
  end #omp for
  if (assembliesInBranch is odd)
    Tree[i-1][end] ← Tree[i][end]
end for                                                  ▷ TforceDis: Toc
Q̄_1, Q̄_2 ← Tree[0].ArticulatedForces()
```

Output: Articulated forces $\overline{\mathbf{Q}}_1, \overline{\mathbf{Q}}_2$ for all physical bodies at the leaf level

Algorithm 5 Evaluation of accelerations and reactions forces (HDCA)

Inputs: $\boldsymbol{\alpha}^k, \dot{\boldsymbol{\alpha}}^k, \overline{\mathbf{P}}_1, \overline{\mathbf{Q}}_1$, `Tree`
Global variables: `Nbodies, Nthreads, Ntiers, input`
Set: in equations (3.39), (3.42): $\mathbf{c}_j := \mathbf{W}_j(\boldsymbol{\zeta}_{10}^B - \boldsymbol{\zeta}_{20}^A - \mathbf{c}_{\text{bias}})$ (4.40)

Numerical procedure:

$\mathbf{q}^k, \mathbf{v}^k \leftarrow$ `mapToAbsoluteVariables(`$\boldsymbol{\alpha}^k, \dot{\boldsymbol{\alpha}}^k$`)` (3.20), (3.21)
$\mathbf{P}_1, \mathbf{P}_2 \leftarrow$ `calculatePhysMomenta(`$\overline{\mathbf{P}}_1$`)` (3.29)
$\mathbf{F}_1, \mathbf{F}_2 \leftarrow$ `computeForcesForLeafBodies(`$\overline{\mathbf{Q}}_1$`)` (3.46: $\mathbf{Q}_1^A = \mathbf{F}_1^A$)
Leaf bodies' coefficients calculations
`#omp for(i=0; i < Nbodies; ++i)` ▷ `TaccLeaves`: Tic
  `Tree[0]` $\leftarrow$ `computeLeafBodiesCoeffs(`$\mathbf{q}^k, \mathbf{v}^k, \mathbf{F}_{1,2}^i, \mathbf{P}_{1,2}^i$`, input)` (4.50)
`end #omp for` ▷ `TaccLeaves`: Toc

Recursion from leaves to root node
`for (i=1; i < Ntiers; ++i)` ▷ `TaccAsm`: Tic
  `upBranch` $\leftarrow$ `Tree[i-1]  /* upperBranch */`
  `assembliesInBranch` $\leftarrow$ `readNoAssemblies(i, Nbodies)`
  `#omp for(j=0; j < assembliesInBranch; ++j)`
    `Tree[i]` $\leftarrow$ `assembleEtaDot(upBranch[2j], upBranch[2j+1])` (3.42)
  `end #omp for`
  `if (assembliesInBranch is odd)`
    `Tree[i][end]` $\leftarrow$ `upperBranch[end]`
`end for` ▷ `TaccAsm`: Toc

Backward recursion from root node to leaf bodies
`Tree[end][0].connectBaseBody()` (3.44)
`for (i=Ntiers-2; i > 0; --i)` ▷ `TaccDis`: Tic
  `assembliesInBranch` $\leftarrow$ `readNoAssemblies(i, Nbodies)`
  `#omp for(j=0; j < assembliesInBranch; ++j)`
    `Tree[i][j].disassemble()` (3.39)
  `end #omp for`
  `if (assembliesInBranch is odd)`
    `Tree[i-1][end]` $\leftarrow$ `Tree[i][end]`
`end for` ▷ `TaccDis`: Toc

Variable projection
`#omp for (i=0; i < Nbodies; ++i)` ▷ `TaccProj`: Tic
  $\dot{\mathbf{V}}_1^B, \dot{\mathbf{V}}_2^A \leftarrow$ `evaluateAbsAcceleration(Tree[0][i])` (4.46, 4.47)
  $\mathbf{H}_i, \mathbf{D}_i, \dot{\mathbf{H}}_i \leftarrow$ `computeHandD(`$\alpha_i, \dot{\alpha}_i$`, input)` (tab. 3.2)
  $\ddot{\alpha}_i^k \leftarrow$ `projectAcceleration(`$\dot{\mathbf{V}}_1^B, \dot{\mathbf{V}}_2^A, \mathbf{H}_i, \dot{\mathbf{H}}_i$`)` (3.45 *derivative*)
  $\boldsymbol{\lambda}_i^k \leftarrow \mathbf{D}_i^T \cdot$ `Tree[0].`$\mathbf{L}_1^i$ (3.39: $\mathbf{L}_1^i = \mathbf{D}_i\boldsymbol{\lambda}_i$)
`end #omp for` ▷ `TaccProj`: Toc

Output: $\ddot{\boldsymbol{\alpha}}^k, \boldsymbol{\lambda}^k$ at the time interval t_k

LIST OF FIGURES

LIST OF TABLES

DOCUMENTED PUBLICATIONS

Some parts of this thesis have been published in journal articles or as proceedings of recent thematic conferences. The following list contains all academic publications the author wrote in the course of his Ph.D. Research:

Journal papers

- **P. Maciąg** and P. Malczyk and J. Frączek: Joint–coordinate adjoint method for optimal control of multibody systems, *Multibody System Dynamics* (2022), DOI:10.1007/s11044-022-09851-y
- **P. Maciąg** and P. Malczyk and J. Frączek: Hamiltonian direct differentiation and adjoint approaches for multibody system sensitivity analysis, *International Journal for Numerical Methods in Engineering* (2020), DOI:10.1002/nme.6512

Chapters in proceedings

- **P. Maciąg** and P. Malczyk and J. Frączek: Sterowanie przestrzennym wahadłem o dwóch stopniach swobody z wykorzystaniem metody adjoint (Stabilization of a two-degree-of-freedom spatial pendulum using the adjoint method), 16. Polish Robotics Conference (KKR) *Prace Naukowe. Elektronika* (2022), t. 197, red. Cezary Zieliński, Alicja Mazur, ISBN 978-83-8156-410-6, ss. 181-190
- **P. Maciąg** and P. Malczyk and J. Frączek: The Discrete Hamiltonian-Based Adjoint Method for Some Optimization Problems in Multibody Dynamics, Multibody Dynamics 2019 - Proceedings of the 9th ECCOMAS Thematic Conference on Multibody Dynamics, *Computational Methods in Applied Sciences* (2020), DOI:10.1007/978-3-030-23132-3_43
- **P. Maciąg** and P. Malczyk and J. Frączek: Direct sensitivity analysis of planar multibody systems in the Hamiltonian framework, *Advances in Mechanism and Machine Science* (2019), DOI:10.1007/978-3-030-20131-9_305

- **P. Maciąg** and P. Malczyk and J. Frączek: Optimal Design of Multibody Systems Using the Adjoint Method, Dynamical Systems in Applications, *Springer Proceedings in Mathematics and Statistics*, vol. 249 (2018), DOI:10.1007/978-3-319-96601-4_22

- **P. Maciąg** and P. Malczyk and J. Frączek: Kinematic Synthesis of Planar Robotic Systems, Proc. International Interdisciplinary PhD Workshop: IIPhDW 2017 (2017), Łódź University of Technology Press

Reviewed abstracts

- **P. Maciąg** and P. Malczyk and J. Frączek: Adjoint-based feedforward control of two-degree-of-freedom planar robot, *Book of Abstracts of the 11th ECCOMAS Thematic Conference on Multibody Dynamics* (2023), Lisbon, Instituto Superior Técnico (abstract accepted)

- **P. Maciąg** and P. Malczyk and J. Frączek: Optimal Design of Multibody Systems Using Independent Adjoint Variables, *Book of Abstracts of the 10th ECCOMAS Thematic Conference on Multibody Dynamics* (2021), Budapest, Budapest University of Technology and Economics

- **P. Maciąg** and P. Malczyk and J. Frączek: Parameter identification of a five-bar multibody system using the Hamiltonian-based adjoint method, *Book of abstracts of the 1st International Conference on Machine Design* (2021), Quântica Editora, Lda.

- P. Malczyk and **P. Maciąg** and J. Frączek: Divide and conquer algorithm for optimal control of multibody systems in the Hamiltonian framework, Multibody Dynamics 2019 - Proceedings of the 9th ECCOMAS Thematic Conference on Multibody Dynamics, *Computational Methods in Applied Sciences* (2020), Springer

REFERENCES

[1] K. Ahnert et al. “Odeint – Solving Ordinary Differential Equations in c++”. In: *AIP Conference Proceedings*. AIP, 2011. DOI: 10.1063/1.3637934.

[2] K. S. Anderson and Y. Hsu. “Analytical fully-recursive sensitivity analysis for multibody dynamic chain systems”. In: *Multibody System Dynamics* 8.1 (2002).

[3] K. J. Åström and R. M. Murray. “Feedback systems”. In: *Princeton University* (2008).

[4] M. Athans and P. L. Falb. “Optimal control: an introduction to the theory and its applications”. In: *Courier Corporation* (2013).

[5] G. Bastos et al. “Inverse dynamics of serial and parallel underactuated multibody systems using a DAE optimal control approach”. In: *Multibody System Dynamics* 30.3 (Mar. 2013). DOI: 10.1007/s11044-013-9361-z.

[6] E. Bayo et al. “A modified lagrangian formulation for the dynamic analysis of constrained mechanical systems”. In: *Computer Methods in Applied Mechanics and Engineering* 71.2 (1988). DOI: 10.1016/0045-7825(88)90085-0.

[7] J. M. Bern et al. “Trajectory Optimization for Cable-Driven Soft Robot Locomotion”. In: *Robotics: Science and Systems* (2019). DOI: 10.15607/RSS.2019.XV.052.

[8] D. Bestle and P. Eberhard. “Analyzing and optimizing multibody systems”. In: *Mechanics of structures and machines* 20 (1992). DOI: 10.1080/08905459208905161.

[9] P. Betsch and R. Siebert. “Rigid body dynamics in terms of quaternions: Hamiltonian formulation and conserving numerical integration”. In: *International Journal for Numerical Methods in Engineering* 79.4 (July 2009). DOI: 10.1002/nme.2586.

[10] P. Betsch et al. “Numerical integration of discrete mechanical systems with mixed holonomic and control constraints”. In: *Journal of Mechanical Science and Technology* 23.4 (Apr. 2009). DOI: 10.1007/s12206-009-0331-6.

[11] J. T. Betts. *Practical methods for optimal control and estimation using nonlinear programming*. Vol. 19. Siam, 2010.

[12] K. D. Bhalerao et al. “An efficient direct differentiation approach for sensitivity analysis of flexible multibody systems”. In: *Multibody System Dynamics* 23.2 (Feb. 2010). ISSN: 1573-272X. DOI: 10.1007/s11044-009-9176-0.

[13] K. D. Bhalerao et al. “An efficient parallel dynamics algorithm for simulation of large articulated robotic systems”. In: *Mechanism and Machine Theory* 53 (July 2012). DOI: 10.1016/j.mechmachtheory.2012.03.001.

[14] N. D. Bianco et al. “Comparison of direct and indirect methods for minimum lap time optimal control problems”. In: *Vehicle System Dynamics* 57.5 (June 2018). DOI: 10.1080/00423114.2018.1480048.

[15] W. Blajer and K. Kołodziejczyk. “A Geometric Approach to Solving Problems of Control Constraints: Theory and a DAE Framework”. In: *Multibody System Dynamics* 11.4 (May 2004). DOI: 10.1023/b:mubo.0000040800.40045.51.

[16] M. Borri et al. “The Embedded Projection Method: A general index reduction procedure for constrained system dynamics”. In: *Computer Methods in Applied Mechanics and Engineering* 195 (Oct. 2006). DOI: 10.1016/j.cma.2005.03.010.

[17] K. Brenan et al. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. Society for Industrial and Applied Mathematics, 1995. DOI: 10.1137/1.9781611971224.

[18] O. Bruls and P. Eberhard. “Sensitivity analysis for dynamic mechanical systems with finite rotations”. In: *International Journal for Numerical Methods in Engineering* 74.13 (2008). DOI: 10.1002/nme.2232.

[19] R. Budhiraja et al. “Differential Dynamic Programming for Multi-Phase Rigid Contact Dynamics”. In: *2018 IEEE-RAS 18th International Conference on Humanoid Robots*. IEEE, Nov. 2018. DOI: 10.1109/humanoids.2018.8624925.

[20] A. Burghardt et al. “Modeling of dynamics of cooperating wheeled mobile robots”. In: *Journal of Theoretical and Applied Mechanics* (Sept. 2021). DOI: 10.15632/jtam-pl/141668.

[21] *C++ OpenMP Parallel For Loop – Alternatives to std::vector*. URL: https://stackoverflow.com/a/18671256.

[22] A. Callejo et al. “Sensitivity-Based, Multi-Objective Design of Vehicle Suspension Systems”. In: *Journal of Computational and Nonlinear Dynamics* 10.3 (May 2015). DOI: 10.1115/1.4028858.

[23] A. Callejo et al. “Adjoint method for the sensitivity analysis of composite beam cross-sections”. In: *Computers & Structures* 213 (Mar. 2019). DOI: 10.1016/j.compstruc.2018.12.004.

[24] Y. Cao et al. “Adjoint sensitivity analysis for differential-algebraic equations: algorithms and software”. In: *Journal of Computational and Applied Mathematics* 149.1 (2002). ISSN: 0377-0427. DOI: https://doi.org/10.1016/S0377-0427(02)00528-9.

[25] J. Carpentier et al. “The Pinocchio C++ library: A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives”. In: *International Symposium on System Integration*. IEEE. 2019.

[26] J. M. Carson et al. “The SPLICE Project: Continuing NASA Development of GN&C Technologies for Safe and Precise Landing”. In: *AIAA Scitech 2019 Forum*. AIAA, Jan. 2019. DOI: 10.2514/6.2019-0660.

[27] K. Chadaj et al. “Efficient parallel formulation for dynamics simulation of large articulated robotic systems”. In: *2015 20th International Conference on Methods and Models in Automation and Robotics (MMAR)* (Aug. 2015). DOI: 10.1109/MMAR.2015.7283916.

[28] K. Chadaj et al. “A parallel Hamiltonian formulation for forward dynamics of closed-loop multibody systems”. In: *Multibody System Dynamics* 39.1 (Jan. 2017). DOI: 10.1007/s11044-016-9531-x.

[29] K. Chadaj et al. “A Parallel Recursive Hamiltonian Algorithm for Forward Dynamics of Serial Kinematic Chains”. In: *IEEE Transactions on Robotics* 33.3 (2017). DOI: 10.1109/TRO.2017.2654507.

[30] C. Chang and P. Nikravesh. “Optimal design of mechanical systems with constraint violation stabilization method”. English (US). In: *American Society of Mechanical Engineers* (1985).

[31] S. Corner et al. “Adjoint sensitivity analysis of hybrid multibody dynamical systems”. In: *Multibody System Dynamics* 49 (Feb. 2020). DOI: 10.1007/s11044-020-09726-0.

[32] J. G. De Jalon and E. Bayo. *Kinematic and dynamic simulation of multibody systems: the real-time challenge*. Springer-Verlag New York, Inc., 1994. DOI: 10.1007/978-1-4612-2600-0.

[33] R. Desai et al. “Interactive Co-Design of Form and Function for Legged Robots using the Adjoint Method”. In: *arXiv* (Jan. 1, 2018). arXiv: 1801.00385v2 [cs.RO].

[34] J. Dias and M. Pereira. “Sensitivity analysis of rigid-flexible multibody systems”. In: *Multibody System Dynamics* 1.3 (1997).

[35] J. Ding et al. “Parameter identification of multibody systems based on second order sensitivity analysis”. In: *International Journal of Non-Linear Mechanics* 47.10 (Dec. 2012). DOI: 10.1016/j.ijnonlinmec.2011.09.009.

[36] D. Dopico et al. “Direct and Adjoint Sensitivity Analysis of Ordinary Differential Equation Multibody Formulations”. In: *Journal of Computational and Nonlinear Dynamics* 10.1 (2014). DOI: 10.1115/1.4026492.

[37] D. Dopico et al. “Direct sensitivity analysis of multibody systems with holonomic and nonholonomic constraints via an index-3 augmented Lagrangian formulation with projections”. In: *Nonlinear Dynamics* 93.4 (Sept. 2018). ISSN: 1573-269X. DOI: 10.1007/s11071-018-4306-y.

[38] D. Dopico et al. *MBSLIM: Multibody Systems in Laboratorio de Ingeniería Mecanica*. 2009-2016. URL: http://lim.ii.udc.es/MBSLIM.

[39] I. Duleba and J. Sasiadek. “Nonholonomic motion planning based on Newton algorithm with energy optimization”. In: *IEEE Transactions on Control Systems Technology* 11.3 (May 2003). DOI: 10.1109/tcst.2003.810394.

[40] A. Dürrbaum et al. “Comparison of Automatic and Symbolic Differentiation in Mathematical Modeling and Computer Simulation of Rigid-Body Systems”. In: *Multibody System Dynamics* 7.4 (2002). DOI: 10.1023/a:1015523018029.

[41] P. Eichmeir et al. “The Adjoint Method for Time-Optimal Control Problems”. In: *Journal of Computational and Nonlinear Dynamics* 16.2 (Nov. 2020). DOI: 10.1115/1.4048808.

[42] R. Featherstone. “A Divide-and-Conquer Articulated-Body Algorithm for Parallel O(log(n)) Calculation of Rigid-Body Dynamics. Part 2: Trees, Loops, and Accuracy”. In: *The International Journal of Robotics Research* 18.9 (Sept. 1999). DOI: 10.1177/02783649922066628.

[43] R. Featherstone. *Rigid Body Dynamics Algorithms*. Springer US, 2008. DOI: 10.1007/978-1-4899-7560-7.

[44] A. Fijany et al. “Parallel O(log N) algorithms for the computation of manipulator forward dynamics”. In: *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*. IEEE Comput. Soc. Press. DOI: 10.1109/robot.1994.351368.

[45] I. Gelfand and S. Fomin. “Calculus of Variations Prentice-Hall”. In: *Englewood Cliffs, NJ* (1963).

[46] M. B. Giles and N. A. Pierce. “An Introduction to the Adjoint Approach to Design”. In: *Flow, Turbulence and Combustion* 65.3/4 (2000). DOI: 10.1023/a:1011430410075.

[47] S. Glaser et al. “Maneuver-Based Trajectory Planning for Highly Autonomous Vehicles on Real Road With Traffic and Driver Interaction”. In: *IEEE Transactions on Intelligent Transportation Systems* 11.3 (Sept. 2010). DOI: 10.1109/tits.2010.2046037.

[48] H. H. Goldstine. *A History of the Calculus of Variations from the 17th through the 19th Century*. Vol. 5. Springer Science & Business Media, 2012.

[49] Q. Gong et al. “Pseudospectral Optimal Control for Military and Industrial Applications”. In: *2007 46th IEEE Conference on Decision and Control*. IEEE, 2007. DOI: 10.1109/cdc.2007.4435052.

[50] F. González et al. “Behaviour of augmented Lagrangian and Hamiltonian methods for multibody dynamics in the proximity of singular configurations”. In: *Nonlinear Dynamics* 85.3 (2016). DOI: 10.1007/s11071-016-2774-5.

[51] H. Górecki. *Optimization and Control of Dynamic Systems*. Springer International Publishing, 2018. DOI: 10.1007/978-3-319-62646-8.

[52] G. Guennebaud and B. Jacob. *Eigen v3*. 2010. URL: https://eigen.tuxfamily.org.

[53] E. J. Haug et al. “Design Sensitivity Analysis of Large-Scale Constrained Dynamic Mechanical Systems”. In: *J. Mech., Trans., and Automation* 2.106 (1984). DOI: 10.1115/1.3258573.

[54] E. Haug. “An index 0 Differential-Algebraic equation formulation for multibody dynamics: Holonomic constraints”. In: *Mechanics Based Design of Structures and Machines* 45.4 (Oct. 2016). DOI: 10.1080/15397734.2016.1246370.

[55] E. J. Haug. *Computer aided kinematics and dynamics of mechanical systems*. Vol. 1. Allyn and Bacon Boston, 1989.

[56] E. J. Haug. “Computer-aided kinematics and dynamics of mechanical systems”. In: *Research Gate* (2021).

[57] E. J. Haug and J. S. Arora. “Design sensitivity analysis of elastic mechanical systems”. In: *Computer Methods in Applied Mechanics and Engineering* 15.1 (July 1978). DOI: 10.1016/0045-7825(78)90004-x.

[58] A. Held et al. “Structural sensitivity analysis of flexible multibody systems modeled with the floating frame of reference approach using the adjoint variable method”. In: *Multibody System Dynamics* 40.3 (Sept. 2016). DOI: 10.1007/s11044-016-9540-9.

[59] S. L. Herbert et al. “FaSTrack: A modular framework for fast and guaranteed safe motion planning”. In: IEEE, Dec. 2017. DOI: 10.1109/cdc.2017.8263867.

[60] A. C. Hindmarsh et al. “SUNDIALS”. In: *ACM Transactions on Mathematical Software* 31.3 (Sept. 2005). DOI: 10.1145/1089014.1089020.

[61] T. A. Howell et al. “Dojo: A Differentiable Simulator for Robotics”. In: *arXiv preprint* (2022). URL: https://arxiv.org/abs/2203.00806.

[62] T. A. Howell et al. “ALTRO: A Fast Solver for Constrained Trajectory Optimization”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2019. DOI: 10.1109/iros40897.2019.8967788.

[63] A. Jain and G. Rodrigues. “Sensitivity analysis of multibody systems using spatial operators”. In: *Proceedings of the international conference on method and models in automation and robotics (MMAR), Poland* (2000).

[64] A. Jain. *Robot and multibody dynamics: analysis and algorithms*. Springer Science & Business Media, 2010.

[65] G. Kennedy and K. Boopathy. “A Scalable Adjoint Method for Coupled Flexible Multibody Dynamics”. In: *57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. American Institute of Aeronautics and Astronautics, Jan. 2016. DOI: 10.2514/6.2016-1907.

[66] D. E. Kirk. *Optimal control theory: an introduction*. Courier Corporation, 2004.

[67] S. Kraft et al. “Parameter identification of multi-body railway vehicle models – Application of the adjoint state approach”. In: *Mechanical Systems and Signal Processing* 80 (Dec. 2016). DOI: 10.1016/j.ymssp.2016.04.037.

[68] M. Kudruss et al. “Efficient Derivative Evaluation for Rigid-Body Dynamics Based on Recursive Algorithms Subject to Kinematic and Loop Constraints”. In: *IEEE Control Systems Letters* 3.3 (July 2019). DOI: 10.1109/lcsys.2019.2914338.

[69] H. Lankarani and P. Nikravesh. “Application of the canonical equations of motion in problems of constrained multibody systems with intermittent motion”. English (US). In: *American Society of Mechanical Engineers, Design Engineering Division* 14 (1988).

[70] T. Lauß. “Optimal Control of Multibody Systems using the Adjoint Variable Approach”. PhD thesis. Institut für Mechanik und Mechatronik, 2019.

[71] T. Lauß et al. “The discrete adjoint gradient computation for optimization problems in multibody dynamics”. In: *Journal of Computational and Nonlinear Dynamics* 12.3 (2017). DOI: https://doi.org/10.1115/1.4035197.

[72] T. Lauß et al. “The discrete adjoint method for parameter identification in multibody system dynamics”. In: *Multibody System Dynamics* 42.4 (Nov. 2017). DOI: 10.1007/s11044-017-9600-9.

[73] S. Li and L. Petzold. “Software and algorithms for sensitivity analysis of large-scale differential algebraic systems”. In: *Journal of Computational and Applied Mathematics* 125.1-2 (Dec. 2000). DOI: 10.1016/s0377-0427(00)00464-7.

[74] S. Li et al. “Sensitivity analysis of differential–algebraic equations: A comparison of methods on a special problem”. In: *Applied Numerical Mathematics* 32.2 (Feb. 2000). DOI: 10.1016/s0168-9274(99)00020-3.

[75] D. Lichtenecker et al. “On the use of adjoint gradients for time-optimal control problems regarding a discrete control parameterization”. In: *Multibody System Dynamics* (Apr. 2023). DOI: 10.1007/s11044-023-09898-5.

[76] S. Liu and T. Bewley. “Adjoint-based system identification and feedforward control optimization in automotive powertrain subsystems”. In: *Proceedings of the American Control Conference*. IEEE, 2003. DOI: 10.1109/acc.2003.1243463.

[77] P. Maciąg et al. “Optimal Design of Multibody Systems Using the Adjoint Method”. In: *Dynamical Systems Theory and Applications* (2017). DOI: 10.1007/978-3-319-96601-4_22.

[78] P. Maciąg et al. “Direct sensitivity analysis of planar multibody systems in the Hamiltonian framework”. In: *Advances in Mechanism and Machine Science* (2019). Ed. by T. Uhl. DOI: 10.1007/978-3-030-20131-9_305.

[79] P. Maciąg et al. “Hamiltonian direct differentiation and adjoint approaches for multibody system sensitivity analysis”. In: *International Journal for Numerical Methods in Engineering* 121.22 (Aug. 2020). DOI: 10.1002/nme.6512.

[80] P. Maciąg et al. “The Discrete Hamiltonian-Based Adjoint Method for Some Optimization Problems in Multibody Dynamics”. In: *Multibody Dynamics 2019* (2020). Ed. by A. Kecskeméthy and F. Geu Flores. DOI: 10.1007/978-3-030-23132-3_43.

[81] P. Maciąg et al. “Joint–coordinate adjoint method for optimal control of multibody systems”. In: *Multibody System Dynamics* 56.4 (Nov. 2022). DOI: 10.1007/s11044-022-09851-y.

[82] G. Maloisel et al. “Singularity-Aware Design Optimization for Multi-Degree-of-Freedom Spatial Linkages”. In: *IEEE Robotics and Automation Letters* 6.4 (Oct. 2021). DOI: 10.1109/lra.2021.3095043.

[83] D. Malyuta et al. *Advances in Trajectory Optimization for Space Vehicle Control*. 2021. arXiv: 2108.02335 [math.OC].

[84] J. R. R. A. Martins et al. “The complex-step derivative approximation”. In: *Transactions on Mathematical Software* 29.3 (Sept. 2003). DOI: 10.1145/838250.838251.

[85] J. R. Martins and J. T. Hwang. "Review and unification of methods for computing derivatives of multidisciplinary computational models". In: *AIAA journal* 51.11 (2013).

[86] P. Masarati et al. "Computational aspects and recent improvements in the open-source multibody analysis software "MBDyn"". In: *Multibody dynamics* (2005). URL: https://www.mbdyn.org/.

[87] R. M. Mukherjee et al. "A divide-and-conquer direct differentiation approach for multibody system sensitivity analysis". In: *Structural and Multidisciplinary Optimization* 35.5 (2008). DOI: 10.1007/s00158-007-0142-2.

[88] K. R. Muske and J. B. Rawlings. "Model predictive control with linear models". In: *AIChE Journal* 39.2 (Feb. 1993). DOI: 10.1002/aic.690390208.

[89] K. Nachbagauer et al. "FreeDyn–a multibody simulation research code". In: *Proc. of 11th World Congress on Comput. Mech.(WCCM), Barcelona, Spain*. 2015. URL: http://www.freedyn.at/.

[90] K. Nachbagauer et al. "The Use of the Adjoint Method for Solving Typical Optimization Problems in Multibody Dynamics". In: *Journal of Computational and Nonlinear Dynamics* 10.6 (2015). DOI: 10.1115/1.4028417.

[91] J. Naudet. "Forward dynamics of multibody systems: a recursive Hamiltonian approach". In: *Faculteit Ingenieurswetenschappen, Vrije Universiteit Brussel* (2005).

[92] J. Naudet et al. "Forward Dynamics of Open-Loop Multibody Mechanisms Using an Efficient Recursive Algorithm Based on Canonical Momenta". In: *Multibody System Dynamics* 10.1 (2003). DOI: 10.1023/a:1024509904612.

[93] M. Neunert et al. "Fast nonlinear Model Predictive Control for unified trajectory optimization and tracking". In: IEEE, May 2016. DOI: 10.1109/icra.2016.7487274.

[94] J. N. Nganga and P. M. Wensing. "Accelerating Second-Order Differential Dynamic Programming for Rigid-Body Systems". In: *IEEE Robotics and Automation Letters* 6.4 (2021). DOI: 10.1109/LRA.2021.3098928.

[95] P. E. Nikravesh. *Computer-aided analysis of mechanical systems*. Prentice-Hall, Inc., 1988.

[96] P. E. Nikravesh. "Systematic reduction of multibody equations of motion to a minimal set". In: *International Journal of Non-Linear Mechanics* (1990).

[97] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer New York, 2006. DOI: 10.1007/978-0-387-40065-5.

[98] S. Oberpeilsteiner et al. "Optimal input design for multibody systems by using an extended adjoint approach". In: *Multibody System Dynamics* 40.1 (Oct. 2016). DOI: 10.1007/s11044-016-9541-8.

[99] N. V. Orlandea. "Multibody Systems History of ADAMS". In: *Journal of Computational and Nonlinear Dynamics* 11.6 (Aug. 2016). DOI: 10.1115/1.4034296.

[100] *Parallel cumulative (prefix) sums in OpenMP: communicating values between threads*. URL: https://stackoverflow.com/a/65681828.

[101] M. A. Patterson and A. V. Rao. "GPOPS-II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems". In: *ACM Transactions on Mathematical Software* 41.1 (Oct. 2014). DOI: 10.1145/2558904.

[102] U. Phutane et al. "Optimal control simulations of two-finger grasps". In: *Mechanism and Machine Theory* 167 (Jan. 2022). DOI: 10.1016/j.mechmachtheory.2021.104508.

[103] M. Pikuliński and P. Malczyk. "On Handling Discontinuities in Adjoint-based Optimal Control of Multibody Systems". In: *26th International Conference on Methods and Models in Automation and Robotics (MMAR)*. 2022. DOI: 10.1109/MMAR55195.2022.9874268.

[104] M. Pikuliński and P. Malczyk. "Adjoint method for optimal control of multibody systems in the Hamiltonian setting". In: *Mechanism and Machine Theory* 166 (2021). DOI: 10.1016/j.mechmachtheory.2021.104473.

[105] I. M. Ross and C. N. D'Souza. "Hybrid Optimal Control Framework for Mission Planning". In: *Journal of Guidance, Control, and Dynamics* 28.4 (July 2005). DOI: 10.2514/1.8285.

[106] S. Schneider and P. Betsch. "On mechanical DAE systems within the framework of optimal control". In: *PAMM* 19.1 (Nov. 2019). DOI: 10.1002/pamm.201900431.

[107] R. Seifried. *Dynamics of Underactuated Multibody Systems*. Springer International Publishing, 2014. DOI: 10.1007/978-3-319-01228-5.

[108] R. Serban and E. J. Haug. "Kinematic and Kinetic Derivatives in Multibody System Analysis". In: *Journal of Structural Mechanics* 26.2 (1998).

[109] R. Serban and A. Hindmarsh. "Cvodes, the sensitivity-enabled ode solver in sundials". In: 5th International Conference on Multibody Systems, Nonlinear Dynamics, and Control. Jan. 2005. DOI: 10.1115/DETC2005-85597.

[110] R. Serban and A. Recuero. "Sensitivity Analysis for Hybrid Systems and Systems With Memory". In: *Journal of Computational and Nonlinear Dynamics* 14.9 (July 2019). DOI: 10.1115/1.4044028.

[111] R. Serban et al. "Chrono::Vehicle: template-based ground vehicle modelling and simulation". In: *International Journal of Vehicle Performance* 5.1 (2019). DOI: 10.1504/ijvp.2019.097096.

[112] A. A. Shabana. "An Important Chapter in the History of Multibody System Dynamics". In: *Journal of Computational and Nonlinear Dynamics* 11.6 (Sept. 2016). DOI: 10.1115/1.4034295.

[113] S. Singh et al. "Efficient Analytical Derivatives of Rigid-Body Dynamics Using Spatial Vector Algebra". In: *IEEE Robotics and Automation Letters* 7.2 (Apr. 2022). DOI: 10.1109/lra.2022.3141194.

[114] G. A. Sohl and J. E. Bobrow. "A Recursive Multibody Dynamics and Sensitivity Algorithm for Branched Kinematic Chains". In: *Journal of Dynamic Systems, Measurement, and Control* 123.3 (July 2000). DOI: 10.1115/1.1376121.

[115] W. Steiner and S. Reichl. "The Optimal Control Approach to Dynamical Inverse Problems". In: *Journal of Dynamic Systems, Measurement, and Control* 134.2 (Jan. 2012). DOI: 10.1115/1.4005365.

[116] J. G. Steven. *The NLopt nonlinear-optimization package*. URL: https://nlopt.readthedocs.io.

[117] M. Szumowski et al. "Preview Control applied for humanoid robot motion generation". In: *Archives of Control Sciences* (2019). DOI: 10.24425/ACS.2019.127526.

[118] T. Takahashi et al. "Computational Design of Statically Balanced Planar Spring Mechanisms". In: *IEEE Robotics and Automation Letters* 4.4 (Oct. 2019). DOI: 10.1109/lra.2019.2929984.

[119] K. Tchoń and J. Ratajczak. "Dynamically consistent Jacobian inverse for nonholonomic robotic systems". In: *Nonlinear Dynamics* 85.1 (Feb. 2016). DOI: 10.1007/s11071-016-2672-x.

[120] E. Todorov et al. "MuJoCo: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2012. DOI: 10.1109/iros.2012.6386109.

[121] S. Turno and P. Malczyk. "FPGA acceleration of planar multibody dynamics simulations in the Hamiltonian–based divide–and–conquer framework". In: *Multibody System Dynamics* 57.1 (Dec. 2022). DOI: 10.1007/s11044-022-09860-x.

[122] Á. L. Varela. "Sensitivity analysis and optimization of the dynamics of multibody systems using analytical gradient based methods". PhD thesis. Escuela Politécnica Superior de Ingenieria de Ferrol, 2022.

[123] L. Weiwei and T. Emanuel. "Iterative Linear Quadratic Regulator Design For Nonlinear Biological Movement Systems". In: *Proceedings of the First International Conference on Informatics in Control, Automation and Robotics*. SciTePress - Science, 2004. DOI: 10.5220/0001143902220229.

[124] M. Wojtyra and J. Frączek. *Kinematyka układów wieloczłonowych*. Wydawnictwa Naukowo-Techniczne, Warszawa, 2008.

[125] Ł. Woliński. "Implementation of the Adaptive Control Algorithm for the KUKA LWR 4+ Robot". In: *Dynamical Systems in Theoretical Perspective*. Ed. by J. Awrejcewicz. Vol. 248. Springer Proceedings in Mathematics & Statistics. Springer International Publishing, 2018. DOI: 10.1007/978-3-319-96598-7_31.

[126] Ł. Woliński and P. Malczyk. "Dynamic Modeling and Analysis of a Lightweight Robotic Manipulator in Joint Space". In: *Archive of Mechanical Engineering* 62.2 (June 2015). DOI: 10.1515/meceng-2015-0016.

[127] H. Zhang et al. "Discrete Adjoint Sensitivity Analysis of Hybrid Dynamical Systems With Switching". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 64.5 (May 2017). DOI: 10.1109/tcsi.2017.2651683.