

Investigating Advantageous Approaches for Incorporating DAE-Based Multibody Models in Model Predictive Control Problems

Arnim Kargl

University of Stuttgart

Henrik Ebel

`henrik.ebel@lut.fi`

Lappeenranta University of Technology

Peter Eberhard

University of Stuttgart

Research Article

Keywords:

Posted Date: April 21st, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-9439249/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License. [Read Full License](#)

Additional Declarations: No competing interests reported.

Investigating Advantageous Approaches for Incorporating DAE-Based Multibody Models in Model Predictive Control Problems

Arnim Kargl^{*}, Henrik Ebel[#], Peter Eberhard^{*}

^{*} Institute of Engineering
and Computational Mechanics
University of Stuttgart

Pfaffenwaldring 9, 70569 Stuttgart, Germany
[arnim.kargl, peter.eberhard]@itm.uni-stuttgart.de

[#] Department of Mechanical Engineering
LUT University
Yliopistonkatu 34
53850 Lappeenranta, Finland
henrik.ebel@lut.fi

ABSTRACT

Multibody dynamics research has enabled efficient and effective simulation, design optimization, control, and identification of mechanical systems, contributing significantly to applications as diverse as simulation-driven and optimization-based engineering design, robotics, digital-twin technology, and many more. Nowadays, different formulations are in use and their properties and comparative differences are especially well understood for *numerical simulation* purposes. Also, software for multibody modeling often predominantly takes simulation applications into focus. However, efficiency is particularly paramount when using multibody system (MBS) models for *model-based real-time control*, where, in particular, model predictive control (MPC) comes to mind. This contribution investigates different ways of discretizing the higher-index differential-algebraic equations (DAEs) of motion of an MBS model when implementing the MPC's underlying optimal control problem (OCP). In contrast to the widely used reduction to an index-1 DAE in combination with an implicit integration scheme, more direct approaches are proposed, exploiting the ability of nonlinear programming solvers to efficiently handle equality constraints. The proposed OCP implementations are evaluated on a set of example MBS, revealing a superior numerical efficiency, especially with state-of-the-art structure exploiting OCP solvers. The presented approaches are chosen so that they introduce as few additional hyperparameters as possible while allowing as much flexibility as possible in OCP design, and, this way, the obtained results demonstrate that the automatic synthesis of real-time capable MPC controllers from MBS models can nowadays be feasible without arduous per-system tuning or overly specific or limited numerical approaches. However, it is also noted that a prospective, automated workflow from MBS model to MPC controller poses certain requirements for multibody modeling software, motivating a discussion of how such software should be designed to facilitate optimal control, revealing desirable characteristics exceeding what is required for simulation purposes.

1 INTRODUCTION

Model predictive control (MPC) is a very popular feedback-control method that is nowadays successfully implemented across many domains due to its advantageous capability to handle nonlinear system dynamics as well as state and input constraints in a straight-forward fashion. As such, it may be among the most popular approaches from optimal control and is considered as one of the most important control technologies also in industry [31]. Both the theory and the numerical implementation has made significant progress since the 1990s, see, e.g., [18], which has, over time, increased the range of realizable applications, especially to faster time scales and dynamics that have originally been too challenging for the computationally intensive method. Being a model-based control approach, MPC requires an accurate yet fast-to-evaluate model of the system's dynamics, which allows predicting the future system behavior and its sensitivities inside an iterative optimization algorithm. In the case of mechanical systems, (elastic) multibody systems form the perfect match for this inevitable trade-off, modeling the essential physical effects to be controlled with fast evaluation times, compared to, e.g., general nonlinear finite element methods. Still, equations of motion from mechanics even in very basic cases offer some challenges more rarely found in models from other domains such as electrical engineering. In particular, when working with multibody systems (MBS), one often obtains equations of motion in the form of differential-algebraic equations (DAEs) of high differential index. In contrast, from an MPC perspective, most literature assumes system models that are ordinary

differential equations, compare [18, 29]. The control of systems described by differential-algebraic equations (DAEs) is considered much more rarely, which may surprise from a multibody perspective. Therefore, this work investigates and discusses multiple different approaches on how to efficiently include multibody dynamics into OCPs, mentioning both theoretical limitations and practical advice when formulating OCPs using multibody models. Such aspects also include desirable properties of multibody modeling software and workflows, for which the ultimate goal may be an automatic workflow from multibody system model to an efficiently implemented model predictive controller with minimal manual tuning besides encoding the control goals in a cost function and constraints. Some decades ago, realizing this idea was out of reach, but with today’s numerical tools, computation power, and MPC theory, it is imaginable to accomplish with reasonable effort, leveraging and bringing together the efforts made in the multibody and MPC communities. Since the beginning of interest into computerized multibody dynamics, powerful software tools have been developed that not only simulate multibody systems’ equations of motion but that help to formulate them. Today, many commercial, non-commercial, and open-source software products and tools exist, allowing even users not well versed in the underlying mechanics to assemble and simulate models successfully, see, e.g., [25, 24, 15, 35]. Similarly, but often more recently, the application of MPC has been drastically simplified with (fast) MPC approaches and tailored numerical optimization algorithms like *acados* [38], *GRAMPC* [9], and *fatrop* [37], only mentioning some popular recent developments.

Motivated by this background, this article focuses on the fundamental aspects of the step from typical multibody models to real-time capable MPC controllers, in particular investigating the efficient discretization of the MPC’s underlying OCPs including MBS dynamics in the form of higher-index DAEs. Naturally, this is not the first article investigating interrelations between optimal control and multibody dynamics and formalisms. For instance, [7] investigates rigid multibody systems without kinematic loops and compares various model formulations in an optimal-control setup with fixed terminal state and end time, using direct collocation to arrive at a discrete-time OCP. The present article deviates from [7] in some foundational choices. For instance, we investigate settings typical for model predictive control, in particular without a defined final time and final state. Instead, the controller may run ad infinitum, e.g., to asymptotically stabilize an equilibrium point. This real-time feedback-control scenario is particularly interesting and challenging for our purposes as, there, an efficient formulation and speedy numerical solution is essential. Further, we use direct multiple shooting as a discretization approach, as it is arguably presently more common in MPC applications and as it is quite straight-forward to set up and use. Further, we go beyond [7] by also considering a model problem with a flexible body, as well as systems with closed kinematic loops, and by analyzing in greater detail several ways to consider DAEs in the resulting OCPs. In addition, we employ highly adaptable formulation and efficient solution strategies based on more recent tools and solvers, tailored particularly to OCPs as they appear for MPC, so that the results give an up-to-date picture on what is possible with today’s tools depending on the formulation chosen while retaining flexibility in the OCP setup. Another interesting study of multibody formulations in relation to optimal control is given in [33], focusing solely on redundant coordinates, i.e., multibody systems in the form of DAEs in two different variants. Using these formulations, [33] focuses on proposing structure-preserving numerical methods to discretize continuous-time OCPs. Due to this different focus, unlike the present article, [33] does not compare to non-redundant formulations, studies one-time solutions of boundary-value problems instead of real-time feedback control in the form of MPC, and generally has less of a focus on calculation speed and repercussions for multibody-modeling software and interactions with modern OCP solvers tailored to and popular with MPC. Further, [17] raises awareness to the issues faced when using index-3-DAE-based multibody models with MPC, and gives particular attention on how to perform accurate-enough state-estimation as a prerequisite to apply MPC. In this regard, it complements nicely the present article regarding the issue of state estimation, but it does not take into view different formulations of multibody models for MPC, instead focusing on a Baumgarte-stabilized reduced-index form and thus providing no insight how other possible formulations might perform. None of the mentioned articles discusses repercussions on the design of multibody software.

Thus, summarizing the state of the art, to the best of the authors’ knowledge, practical guidelines around MPC with DAEs are rare in the literature, especially for multibody dynamics. As such, there seems to be a lack of studies discussing in detail and in a comparative manner different viable ways of formulating models and associated OCPs in this context, particularly concerning formulation variants where the optimization algorithm takes control of the dynamics’ Lagrange multipliers. To provide novel insights in

this context, the article presents and compares three possible solutions to implement model predictive controllers for (elastic) multibody systems, formulated in terms of differential algebraic equations. First, the classic approach, applying index reduction to obtain an index-1 DAE that can be discretized with implicit integration schemes [19] is mentioned. Due to long computation times and drift effects, this approach is, however, not ideal for MPC applications. Novel insights and better results are obtained using two alternative formulations, which either use the index-3 DAE directly or make use of an index-2 formulation known as the method of Gear, Gupta, and Leimkuhler [14]. Individual advantages and drawbacks of both proposed approaches are discussed with representative examples. For those examples and variants, we also compare the obtained calculation performance when using redundant models in the OCP, where all bodies are joined together by algebraic constraints, to (minimal or generalized-coordinate) model variants with as few constraints as possible. This gives valuable insights when seeking maximum computational efficiency to meet real-time requirements. Alongside the main contribution, the research reveals requirements towards multibody-modeling software as posed by a convenient yet computationally efficient model-to-MPC-controller workflow, exemplifying how to achieve an automatic workflow from multibody modeling to an MPC algorithm implemented on embedded hardware.

The article is structured as follows. Section 2 introduces the studied optimal control problem and the different ways to obtain its discretization in form of nonlinear programs. Section 3 investigates the proposed approaches with a set of example multibody systems. Section 4 discusses higher-level repercussions of the findings, e.g., on the design of multibody modeling software. Finally, Section 5 concludes the work and gives hints on further research directions.

2 OPTIMAL CONTROL PROBLEM FORMULATION

Assuming the reader's familiarity with the equations of motion of a general multibody system, compare [32, Section 5.3], we directly state the continuous-time optimal control problem

$$\underset{\mathbf{u}(\tau; t)}{\text{minimize}} \quad \int_t^{t+T} \ell(\mathbf{x}(\tau; t), \mathbf{u}(\tau; t)) \, d\tau \quad (1a)$$

$$\text{s.t.} \quad \mathbf{x}(\tau; t) = \hat{\mathbf{x}}(t), \quad (1b)$$

$$\mathbf{M}(\mathbf{q}(\tau; t))\ddot{\mathbf{q}}(\tau; t) - \mathbf{C}^\top(\mathbf{q}(\tau; t))\boldsymbol{\lambda}(\tau; t) = \mathbf{h}(\mathbf{q}(\tau; t), \dot{\mathbf{q}}(\tau; t), \mathbf{u}(\tau; t)), \quad (1c)$$

$$\mathbf{c}(\mathbf{q}(\tau; t)) = \mathbf{0}, \quad (1d)$$

$$\mathbf{u}(\tau; t) \in \mathcal{U}. \quad (1e)$$

Therein, the positions $\mathbf{q}(t) \in \mathbb{R}^{n_q}$ together with the velocities $\dot{\mathbf{q}}(t) \in \mathbb{R}^{n_q}$ form the system's state $\mathbf{x}(t) := [\mathbf{q}^\top(t) \quad \dot{\mathbf{q}}^\top(t)]^\top$, the vector $\boldsymbol{\lambda}(t) \in \mathbb{R}^{n_c}$ comprises the Lagrange multipliers associated with the kinematic constraints, and $\mathbf{u}(\cdot; t) : [t, t+T] \rightarrow \mathcal{U} \subseteq \mathbb{R}^{n_u}$ is the control input trajectory to be optimized at time t over the prediction horizon of duration T . The system's mass matrix is given by $\mathbf{M} \in \mathbb{R}^{n_q \times n_q}$, generalized forces are summarized in the vector $\mathbf{h} \in \mathbb{R}^{n_q}$, and $\mathbf{C} \in \mathbb{R}^{n_c \times n_q}$ denotes the Jacobian matrix of the kinematic constraints, i.e., $\mathbf{C} := \partial \mathbf{c} / \partial \mathbf{q}$. The number of coordinates and constraints are n_q and n_c , respectively. Equations (1c) and (1d) together form the index-3 DAE of the multibody system, which is initialized at the beginning of the prediction horizon, usually with an estimate of the system's state, as formulated in Equation (1b). Furthermore, the control input is restricted to the set of admissible inputs $\mathcal{U}$, c.f., Equation (1e), and the control objective is expressed in terms of a general cost function $\ell : \mathbb{R}^{n_x} \times \mathcal{U} \rightarrow \mathbb{R}$, which maps the combined set of system states and inputs to a scalar. In model predictive control applications, the same OCP is repeatedly solved at subsequent time instants, initialized with a different state $\hat{\mathbf{x}}(t)$ as time passes, thereby closing the feedback loop [29]. The procedure is often referred to as a *receding horizon scheme*, as the prediction horizon of length T recedes with each time step. Formally, guaranteeing closed-loop stability can be facilitated through further ingredients, e.g., terminal constraints, which are left out for the discussion, but may be added without loss of generality. As considerations of proving nominal closed loop properties like asymptotic stability are not in the focus of this article's questions of formulation procedures and numerical efficiency, we omit such considerations here and instead refer the reader to encompassing literature on the topic [18, 29].

So far, we have not made any assumption on the set of coordinates that are used to describe the MBS

as, on a superficial level, the coordinate choice does not influence a fundamental discussion. In practice, one might choose a representation in minimal coordinates, in absolute coordinates, or in any other type of redundant coordinates [42, 26, 13]. Phenomenologically, in the stated OCP, in the extreme case of a minimal-coordinates model that also does not have any kinematic loops, there simply would not be any constraints, i.e., $n_c = 0$, and, consequently, OCP constraint (1d) would vanish, as would the term including the vector of Lagrange multipliers from Equation (1c). However, two assumptions on the character of the multibody system and model are presumed in the following. Firstly, we consider so-called ideal multibody systems, namely MBS for which the externally applied forces do not depend on reaction forces [32], which would be the case, e.g., when modeling frictional contact. Such effects are usually neglected in models used for model-based control as they are typically hard to model a priori with persistent accuracy. Thus, they are commonly seen as a disturbance on the system to be compensated by a feedback controller and often motivate the usage of feedback control in the first place. Secondly, the kinematic constraints (1d) are assumed to be holonomic and the set of constraint equations is assumed to be non-redundant such that the Jacobian matrix C has full row rank. Still, an extension to nonholonomic systems, which are of wide interest in the robotics community [8], is readily possible but may require special care when designing cost functions [30]. From an optimal control perspective, we silently assume that the OCP with its ingredients like the cost function is well formulated so that the OCP admits a well-defined optimal solution and so that it correctly encodes the control goals, see [18, 2]. Further, we assume that cost functions and constraints are twice continuously differentiable to facilitate practical numerical solution.

In the following, the different considered OCP discretizations, each using a direct multiple shooting method, are explained. For simplicity, we choose the shooting nodes to coincide with the discretization steps of the control input, where the latter is subject to a zero-order hold. Therefore, let $k \in \mathbb{N}_0$ be an index variable connecting discretized signals to continuous time via a constant sampling time δ , e.g., $\mathbf{u}_k := \mathbf{u}(t + \delta k)$. As of tools for the formulation of OCPs, for all numerical examples, the CasADi toolbox [1] is used. Featuring a symbolic engine, automatic differentiation, and interfaces to integration schemes as well as nonlinear programming solvers, it is a widely used tool in the MPC community. In particular, we make use of the interior point optimization algorithm IPOPT [39], if not stated otherwise.

2.1 Discretization of Index-1 DAE with Implicit Integration Scheme

Index reduction is a common approach in multibody simulation due to the fact that many implicit integration schemes do not only support ODEs but also the treatment of index-1 DAEs [19]. Differentiating the constraint equation (1d) twice with respect to time leads to

$$\ddot{\mathbf{c}} = \mathbf{C}\ddot{\mathbf{q}} + \dot{\mathbf{C}}\dot{\mathbf{q}} + \frac{\partial^2 \mathbf{c}}{\partial t^2} = \mathbf{C}\ddot{\mathbf{q}} + \bar{\mathbf{c}}, \quad (2)$$

which, combined with the differential equation (1c), is the index-1 DAE representation of the equations of motion. In the simplest case, an implicit Euler integration scheme is implemented by solving the set of nonlinear equations

$$\mathbf{g}(\mathbf{q}_{k+1}, \dot{\mathbf{q}}_{k+1}, \ddot{\mathbf{q}}_{k+1}, \boldsymbol{\lambda}_k) = \begin{bmatrix} \mathbf{q}_{k+1} - \mathbf{q}_k - \delta \dot{\mathbf{q}}_{k+1} \\ \dot{\mathbf{q}}_{k+1} - \dot{\mathbf{q}}_k - \delta \ddot{\mathbf{q}}_{k+1} \\ \mathbf{M}(\mathbf{q}_{k+1})\ddot{\mathbf{q}}_{k+1} = \mathbf{h}(\mathbf{q}_{k+1}, \dot{\mathbf{q}}_{k+1}, \mathbf{u}_k) + \mathbf{C}^\top(\mathbf{q}_{k+1})\boldsymbol{\lambda}_k \\ \mathbf{C}(\mathbf{q}_{k+1})\ddot{\mathbf{q}}_{k+1} + \bar{\mathbf{c}}(\mathbf{q}_{k+1}, \dot{\mathbf{q}}_{k+1}) \end{bmatrix} = \mathbf{0} \quad (3)$$

with a Newton solver, in practice, e.g., as provided by CasADi. Since all necessary terms are formulated symbolically, the complete integration step

$$\mathcal{I}: (\mathbf{q}_k, \dot{\mathbf{q}}_k; \mathbf{u}_k) \mapsto (\mathbf{q}_{k+1}, \dot{\mathbf{q}}_{k+1}), \quad (4)$$

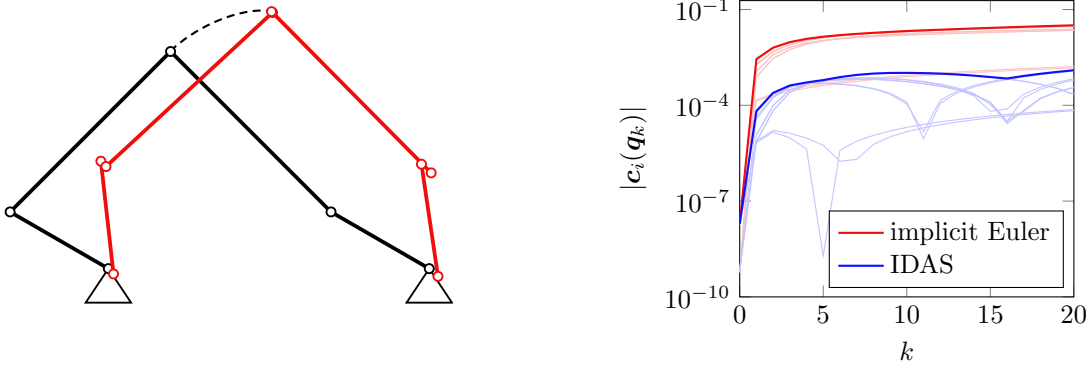


Figure 1: Planar five-bar linkage performing a set point change. On the left, the initial condition as well as the linkage’s final pose at the end of the prediction horizon is visualized, revealing numerical drift effects when using an implicit Euler integration scheme (red). On the right, the quantitative violation of each constraint equation is shown for both the implicit Euler integration scheme (red) and the IDAS integrator (blue) is shown. The maximum constraint violation is highlighted bold.

including the Newton solver, can be automatically differentiated, which gives the necessary gradients when solving the optimal control problem. The complete resulting nonlinear program can be stated as

$$\underset{\mathbf{u}_k, \mathbf{x}_k}{\text{minimize}} \quad \sum_{k=0}^{K-1} \delta \ell(\mathbf{x}_k, \mathbf{u}_k) \, d\tau \quad (5a)$$

$$\text{s.t.} \quad \mathbf{x}_0 = \hat{\mathbf{x}}(t), \quad (5b)$$

$$\mathbf{x}_{k+1} = \mathcal{I}(\mathbf{x}_k; \mathbf{u}_k), \quad k \in \{0, 1, \dots, K-1\}, \quad (5c)$$

$$\mathbf{u}_k \in \mathcal{U}, \quad k \in \{0, 1, \dots, K-1\}. \quad (5d)$$

Thereby, Equation (5a) is one simple way of discretizing the continuous-time cost function, and Equation (5c) makes use of the above-defined integration step over one shooting interval δ . Note that an implicit Euler integration scheme, as it is used here for its simplicity, does not provide any numerical guarantees when applied to DAEs [19]. Hence, the obtained solutions should always be checked for consistency, or a less simple scheme may be employed. For instance, in practice, CasADi offers an interface to the SUNDIALS [20] suite, which provides backward differentiation formulas of higher order with integrated numerical sensitivity propagation.

To give some more insight into what the problems arising from numerical drift mean for optimal control, a planar five-bar linkage is inspected for a single open-loop OCP solution, as it would happen once per time step for MPC. The system consists of two planar double pendulums with two degrees of freedom each, mounted parallel to each other and joining the pendulums’ ends together in a hinge, which forms the system’s end effector and leaves two degrees of freedom of the overall linkage. A single solution of NLP (5) is computed once when using the implicit Euler integration scheme and once when using the SUNDIALS integrator IDAS. As a control task, the end effector shall perform a set-point change, which is depicted on the left-hand side of Figure 1, supplying adequate control torques to the base joints. The resulting constraint violation in the individual joints as well as the maximum constraint violation, as resulting from the index reduction, are visualized on the right-hand side of Figure 1 with each light-shaded line per color corresponding to the violation of one constraint equation, and the one bold line per color marking the maximum error per time step in the horizon per discretization scheme. While it may not fully prevent a model predictive control application, it is clearly visible that numerical drift can be a serious issue with the index-1 DAE approach even across typical MPC prediction horizon lengths. The linkage’s longer links have a length of 1 m and the maximum constraint violation in combination with the simple implicit Euler integration scheme is in the range of multiple centimeters. The left-hand side of Figure 1 displays the linkage at the end of the OCP’s prediction horizon when using the implicit Euler discretization, showing that the numerical drift is readily visible. Using the more accurate IDAS integrator, drift effects are much less severe, i.e., in the range of one millimeter, but other downsides are mentioned below.

Of course, there are many known ways to mitigate these numerical issues, such as additional projection steps or Baumgarte stabilization [27]. However, these measures either require additional, expensive computations or must be carefully tuned for each system. A simple and seemingly universal measure would be to just include the original constraint equations $\mathbf{c}(\mathbf{q}_k) = \mathbf{0}, k \in \{1, 2, \dots, N\}$, into the NLP, recognizing that equality constraints are in principle no issue for a nonlinear optimization algorithm. However, just by counting the number of free variables and comparing it to the number of formulated equality constraints, the problem inevitably becomes overconstrained. Consequently, trying this trivial solution immediately leads to an error, caught in practice already in the setup of the solver, such that it is not an option. The difficulty lies in the fact that the Lagrange multipliers $\boldsymbol{\lambda}$ are capsuled inside the numerical integration scheme $\mathcal{I}$ and are, hence, cannot be directly influenced the optimization algorithm.

Before moving to less naïve discretizations, a brief comment on the computation times, which will be studied in more detail later, is prudent. After all, in the given example, some may argue that the usage of a more involved implicit integration scheme like the IDAS integrator, even without additional drift-mitigation matters, seems accurate enough in practice, even if it does not intrinsically solve the issue of drift. However, such precise integration can come at the cost of a prohibitive computational penalty. When using the IDAS integrator to perform the integration $\mathcal{I}$, the OCP's solution requires multiple seconds on a standard desktop PC, which is due to the involved numerical integration algorithm and numerical sensitivity propagation. As a matter of fact, even when reducing the complexity to the above-described implicit Euler integrator, still multiple hundred milliseconds are required, such that closed-loop control cannot be achieved with the tools used. There are efforts regarding MPC with implicit integration schemes, for instance *acado* [28] or the successor *acados* [38], pushing computation times into the sub-millisecond range, but these specialized algorithms are not only less flexible regarding different OCP ingredients but again have many hyperparameters that may have to be tuned for the individual use case in order to bring reliably accurate OCP solutions and calculation speed together. Hence, in this article, we focus on workflows with as little manual tuning and as much flexibility as possible and, thus, we subsequently do not pursue *acados* or similar tools further in this article.

2.2 Discretization of the Index-3 DAE

Following the discussions of the previous section, we propose to directly discretize the equations of motion, i.e., the index-3 differential-algebraic equation. In essence, the unconstrained system dynamics (1c) is discretized by an arbitrary numerical integration scheme, introducing Lagrange multipliers $\boldsymbol{\lambda}_k, k \in \{0, 1, \dots, K-1\}$, as free optimization variables. Hence, the unconstrained system is influenceable in each integration step both by the input $\mathbf{u}_k$ and the Lagrange multipliers $\boldsymbol{\lambda}_k$, meaning that, in the unconstrained system, the Lagrange multipliers appear structurally in a similar fashion as the input and are optimized by the optimization algorithm like an input. The multipliers are essential to keep the system on the constraint manifold imposed by $\mathbf{c}(\mathbf{q}) = \mathbf{0}$, whereas the inputs can freely steer the system along said constraint manifold. To ensure that the Lagrange multipliers fulfill their goal of keeping the system on the constraint manifold, the constraint (1d) on position level is directly included into the NLP, such that it reads

$$\underset{\mathbf{u}_k, \mathbf{x}_k, \boldsymbol{\lambda}_k}{\text{minimize}} \quad \ell_K(\mathbf{x}_K) + \sum_{k=0}^{K-1} \delta \ell(\mathbf{x}_k, \mathbf{u}_k) \, \mathrm{d}\tau \quad (6a)$$

$$\text{s.t.} \quad \mathbf{x}_0 = \hat{\mathbf{x}}(t), \quad (6b)$$

$$\mathbf{x}_{k+1} = \mathcal{I}(\mathbf{x}_k; \mathbf{u}_k, \boldsymbol{\lambda}_k), \quad k \in \{0, 1, \dots, K-1\}, \quad (6c)$$

$$\mathbf{c}(\mathbf{x}_k) = \mathbf{0}, \quad k \in \{2, 3, \dots, K\}, \quad (6d)$$

$$\mathbf{u}_k \in \mathcal{U}, \quad k \in \{0, 1, \dots, K-1\}. \quad (6e)$$

In contrast to Section 2.1, no assumptions on the numerical integration scheme $\mathcal{I}$ are necessary, which renders the often-used explicit Runge-Kutta integration schemes attractive again. Since the application of model predictive control typically anyway implies fast sampling rates, often even an explicit Euler integrator is sufficiently accurate.

Regarding NLP (6), two aspects need a more detailed explanation. Firstly, note the additional cost term $\ell_K(\mathbf{x}_K)$ in Equation (6a). It is introduced so that $\boldsymbol{\lambda}_{K-1}$, which influences the system state at the end of the prediction horizon, i.e., $\mathbf{x}_K = \mathcal{I}(\mathbf{x}_{K-1}; \mathbf{u}_{K-1}, \boldsymbol{\lambda}_{K-1})$, has an influence on the cost function. Otherwise,

this Lagrange multiplier could be chosen arbitrarily. For optimality, an arbitrarily set λ_{K-1} would indeed not matter, but we observed that the optimization algorithm performs additional, unnecessary iterations in this case. Thus, this is a manifestation of the general advice to formulate OCPs so that they admit a unique optimal solution, at least locally. Secondly, note that, in Equation (6d), the kinematic constraint is enforced for all integration steps but the first one, which is explained with a minimal example in the following.

Consider a block of mass m with position $\mathbf{q} := [r_x \ r_y]^\top$ sliding without friction on a horizontal plane. A horizontally acting force F is used as the control input. The system is modeled with absolute coordinates and can be described by the DAE

$$\begin{bmatrix} m & 0 \\ 0 & m \end{bmatrix} \begin{bmatrix} \ddot{r}_x \\ \ddot{r}_y \end{bmatrix} = \begin{bmatrix} F \\ mg + \lambda \end{bmatrix}, \quad r_y = 0 \quad (7)$$

with the gravitational constant g and λ representing the reaction force between block and plane in the vertical direction. Transferring this equation into state-space form and applying the first step of an explicit Euler integration scheme, starting from a consistent initial condition $\mathbf{x}_0$, leads to

$$\underbrace{\begin{bmatrix} r_{x,1} \\ r_{y,1} \\ \dot{r}_{x,1} \\ \dot{r}_{y,1} \end{bmatrix}}_{\mathbf{x}_1} = \underbrace{\begin{bmatrix} r_{x,0} \\ 0 \\ \dot{r}_{x,0} \\ 0 \end{bmatrix}}_{\mathbf{x}_0} + \delta \underbrace{\begin{bmatrix} \dot{r}_{x,0} \\ 0 \\ F_0/m \\ g + \lambda_0/m \end{bmatrix}}_{\mathbf{f}(\mathbf{x}_0, F_0, \lambda_0)}. \quad (8)$$

The combination of a consistent initial condition, which fulfills not only $\mathbf{c} = \mathbf{0}$, but also $\dot{\mathbf{c}} = \mathbf{0}$, and the system being of relative degree two, if the state r_x is considered as the output, means that the position constraint is automatically satisfied at time step one, i.e., it anyway holds that $r_{y,1} = 0$. Hence, adding the constraint a second time within the optimization problem would lead to a rank deficient Jacobian matrix of the constraints, which in turn would result in numerical issues.

Concerning the implementation of the optimal control problem with the discretization as in Equations (6c) and (6d), the OCP framework needs to accept different constraints in different stages. Formulating the OCP in CasADi or using the OCP interface of the NLP solver fatrop [37], this does not impose limitations. Other popular MPC toolboxes, such as acados [38] or GRAMPC [9] have stricter requirements to the OCP's structure which complicates or even hinders the implementation. Another topic worth a short comment is the use of higher-order explicit Runge-Kutta methods. Since the different stages of these methods evaluate the system dynamics at intermediate time steps, additional Lagrange multipliers associated with those time steps become necessary if the kinematic constraints are also to be fulfilled at said intermediate time steps. This phenomenon as well as the use of implicit integration methods is, however, not described in detail in this contribution, but we note that the approach naturally extends to these cases, as we have studied separately. The examples studied later merely do not require doing so.

2.3 Discretization as Index-2 Formulation

Since the direct approach of incorporating index-3 DAEs into the optimal control problem can face some hurdles with MPC tools, a third alternative is studied, namely a specific kind of index-2 formulation, also known as the method of Gear, Gupta, and Leimkuhler (GGL) [14]. The method performs a reduction of the original DAE to a DAE with a differential index of two, but with additional Lagrange multipliers $\boldsymbol{\mu}(t) \in \mathbb{R}^{n_c}$, such that the original position constraint is fulfilled under mild assumptions [14]. Let $\mathbf{q}(t)$ denote the absolute positions as before and introduce the absolute velocities $\mathbf{v}(t) \in \mathbb{R}^{n_q}$, then the GGL index-2 formulation is stated as

$$\dot{\mathbf{q}}(t) = \mathbf{v}(t) + \mathbf{C}^\top \boldsymbol{\mu}(t), \quad (9a)$$

$$\mathbf{M}\dot{\mathbf{v}}(t) = \mathbf{h} + \mathbf{C}^\top \boldsymbol{\lambda}(t), \quad (9b)$$

$$\mathbf{c} = \mathbf{0}, \quad (9c)$$

$$\mathbf{C}\mathbf{v}(t) = \mathbf{0}. \quad (9d)$$

When referring to state-space representations, the state is consequently defined as $\mathbf{x}(t) := [\mathbf{q}^\top(t) \ \mathbf{v}^\top(t)]^\top$.

The direct multiple shooting discretization of Equation (9) is obtained analogously to the previous section. An explicit Runge-Kutta integration scheme is used to discretize the unconstrained system's dynamics (9a) and (9b), where both λ_k and μ_k , $k \in \{0, 1, \dots, K-1\}$, are free optimization variables. The resulting nonlinear program enforcing both constraints (9c) and (9d) reads

$$\underset{\mathbf{u}_k, \mathbf{x}_k, \lambda_k, \mu_k}{\text{minimize}} \quad \ell_K(\mathbf{x}_K) + \sum_{k=0}^{K-1} \delta \ell(\mathbf{x}_k, \mathbf{u}_k) \, d\tau \quad (10a)$$

$$\text{s.t.} \quad \mathbf{x}_0 = \widehat{\mathbf{x}}(t), \quad (10b)$$

$$\mathbf{x}_{k+1} = \mathcal{I}(\mathbf{x}_k; \mathbf{u}_k, \lambda_k, \mu_k), \quad k \in \{0, 1, \dots, K-1\}, \quad (10c)$$

$$\mathbf{c}(\mathbf{x}_k) = \mathbf{0}, \quad k \in \{1, 2, \dots, K\}, \quad (10d)$$

$$\mathbf{C}(\mathbf{x}_k) \mathbf{v}_k = \mathbf{0}, \quad k \in \{1, 2, \dots, K\}, \quad (10e)$$

$$\mathbf{u}_k \in \mathcal{U}, \quad k \in \{0, 1, \dots, K-1\}. \quad (10f)$$

In contrast to the direct index-3 formulation, no special care has to be taken regarding the kinematic constraint equations since the additional multipliers μ_k act on velocity level, and thus have a direct influence already in the first integration step. Hence, the formulation fits a wider range of MPC frameworks, of course with the drawback of an increased amount of optimization variables as well as constraints, which sometimes has a considerable impact on the computation time, as seen in Section 3.

3 COMPARATIVE STUDY

In this section, the previously introduced approaches to solve optimal control problems including multibody dynamics in differential-algebraic form are evaluated with three exemplary multibody systems. The goal is to demonstrate the efficiency and applicability of the proposed OCP discretizations, not only setting the focus on the computational effort, but also studying the influence of the choice of coordinates describing a multibody system. Furthermore, two exemplary investigations are made, one with the structure-exploiting interior point optimizer fatrop [37], revealing the obtainable speed-up when implementing the NLP directly in C++, and one imposing limits on joint reaction forces by constraints on the Lagrange multipliers.

3.1 Simulation Setup

Before going into detail on the results, the used software toolchain as well as the simulation setup are briefly described, which will later be useful to reflect on multibody software setups facilitating the (automated) setup of optimal control problems. The equations of motion for the upcoming discussions are assumed to be available in symbolic form, enabling efficient manipulation, evaluation, and code export to implement the above-described OCPs. In this sense, the name *symbolic* refers to all parameters, variables, and coordinates being treated as symbols that are later replaced by their actual numeric value before the equations of motion are solved. A workflow that satisfies these requirements is depicted in Figure 2. Central part of the workflow is the multibody system toolbox Neweul-M² [24], successor of NEWEUL [25], which is developed at the Institute of Engineering and Computational Mechanics. A key feature is the symbolic derivation of the equations of motion based on the Newton-Euler equations. Both recursive and iterative formalisms to formulate the kinematics of systems in tree structure are available. Systems with kinematic loops are modeled by formulating the kinematics of the unconstrained system and adequate loop-closing constraints. Systems with algebraic constraint equations are efficiently handled with a coordinate partitioning approach in numerical simulation [23]. Therefore, the software provides an automated export of the equations of motion to Matlab code, which we use together with Matlab's numerical integration algorithms in the following closed-loop simulations, i.e., to simulate the mechanical systems with the system input provided by the model predictive controller. Individual symbolic terms of the equations of motion are exported from Neweul-M² into the symbolic framework of CasADi [1] for the implementation of the NLPs. Most of the following studies use CasADi's interface to the interior point optimization algorithm IPOPT [39] to solve the NLPs. In Section 3.3, interfacing the fatrop solver from C++, CasADi's ability to export efficient C/C++ code is used to derive and export all needed terms, such as gradients and Hessian matrices.

Concerning elastic multibody systems, the floating frame of reference formulation is used, importing flexible bodies to Neweul-M² via standard input data (SID) [34]. In the simplest form, data such as mass and

stiffness coefficients of the elastic bodies are then numerically integrated into the otherwise symbolic equations of motion. A more abstract handling of elastic elements in purely symbolic form is possible [4], but requires evaluating the standard input data during numerical evaluation, which complicates the reuse of the equations of motion. In this contribution, the Matlab version of Morembs [10] is used to generate SID files compatible to Neweul-M². Furthermore, Morembs provides interfaces to popular finite element software as well as various methods for model-order reduction, e.g., modal reduction, Krylov subspace methods, and balanced truncation.

3.2 Comparison of the OCP Formulations in MPC

For comparison, three representative systems are set up using the above-described modeling workflow. Next to the already introduced planar five-bar linkage, a classical delta robot is set up, representing a more complex multibody system with kinematic loops in 3D space. Finally, a serial manipulator with three links is modeled. Its first link consists of a flexible body, allowing to extend the investigations to an elastic multibody system. The flexible link is formulated using Euler-Bernoulli beam elements and modal reduction to one eigenmode is performed for control purposes. Further eigenmodes are neglected in the control model as they already correspond to very high eigenfrequencies, which would require adequately small sampling intervals. The simulation model usually obtains more eigenmodes, aiming for an accurate representation of the real system. However, when using redundant coordinates, this leads to problems with inconsistent initial conditions of the OCP, which is a topic separately discussed in Section 3.5.

Each of the three systems is modeled in two variants, once using generalized coordinates, aiming for a minimum number of coordinates and constraints, and once using a set of redundant, here, absolute Cartesian, coordinates, as they are widely used in practice. In both cases, rotations are described by subsequent rotations around individual coordinate axes as generalized coordinates. Also popular in the context of absolute coordinate formulations are redundant descriptions like Euler parameters, which are not discussed here, since the studied systems are not problematic concerning rotation locking phenomena. Regarding the manipulator's flexible link, the use of absolute or minimal coordinates only concerns the position and orientation of the floating frame of reference. The elastic deformation is, here, described relative to said reference frame, as described in [34].

The main results of the investigation are summarized in Table 1, listing the computation times obtained solving the model predictive controllers' underlying OCPs with the IPOPT solver. For reasons of compa-

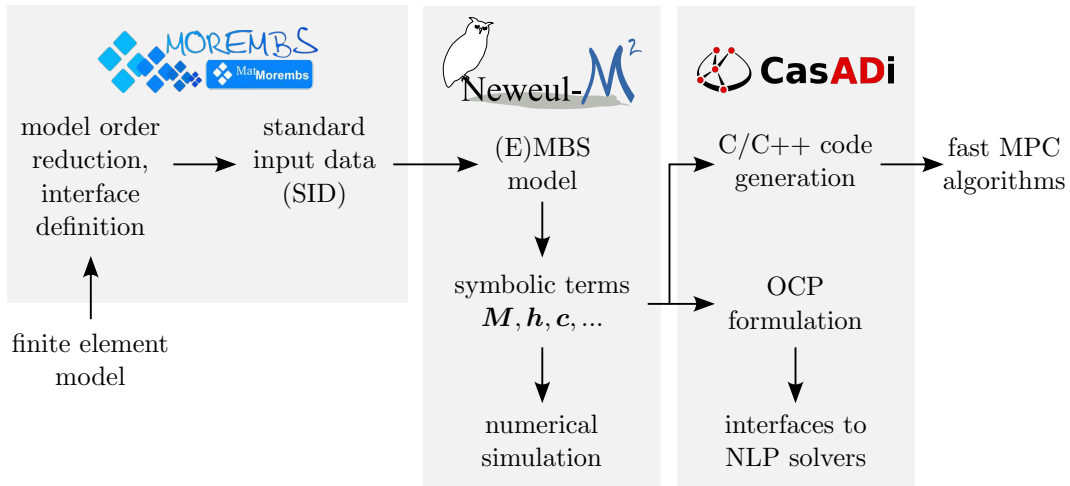


Figure 2: Workflow to obtain nonlinear programs (NLPs) for elastic multibody systems (EMBS), performing model order reduction and generating standard input data (SID) [34] for elastic bodies with Morembs [10]. Multibody system models are set up using the symbolic MBS toolbox Neweul-M² [24]. CasADi [1] is used to formulate optimal control problems and export efficient C/C++ code for further external usage.

rability, the prediction horizon of each optimal control problem is discretized into $K = 20$ discretization intervals. The absolute length in seconds of the prediction horizon is listed as well, although it is not directly relevant for the subsequent discussion. Here, only for the manipulator with a flexible link, a faster sampling rate is chosen since the elastic motion introduces a comparatively high eigenfrequency into the system. Furthermore, as the serial manipulator is a system without kinematic loops, it allows the formulation of a representation in minimal coordinates without any kinematic constraints at all.

Regarding the measured computation times depicted in the table, multiple observations are made. Evidently, reducing the equations of motion to an index-1 DAE and using an implicit integration scheme can work effectively in simple cases, such as the five-bar linkage. However, the computational effort connected to the underlying iterative nonlinear equation solver used within the integration algorithm which is, again, embedded within each iteration of an optimization algorithm, leads to a comparatively high computation time. As already observed earlier for the five-bar linkage, also in the other examples, a drastic increase in computation time is observed using the IDAS integration scheme, arising from the fact that it is a more precise integration algorithm with variable order as well as variable step size. While the IDAS integrator's precision is desirable concerning drift, its computational effort is preventing real-time MPC applications. In our empirical studies, this was still the case even when using rather coarse settings for the algorithm's tolerance parameters. For the manipulator with flexible link, the simple implicit Euler integration scheme with constant step size fails to stabilize the system, which is due to the sampling time of 5 ms, which is still too coarse to reliably cover the system's highest eigenfrequency. Here, the direct implementations of the index-2 or index-3 DAE, respectively, are advantageous, with the original kinematic constraint on position level making the MPC's prediction more robust. Regarding the computation times of these implementations, a system representation in generalized coordinates seems generally preferable. The only exception is the manipulator with flexible link. Since the flexible link is the first one in the chain, the kinematics of the other links depend on the first link's deformation when using a minimal-coordinates formulation, leading to particularly lengthy expressions in the equations of motion, and hence the OCP, that can be time-consuming to evaluate, especially in uncompiled and unoptimized code.

Still, some observations might be surprising at first glance. For simulation purposes, for systems that anyway have a number of position-level kinematic constraints, many modelers might by standard opt for a

Table 1: System properties, number of optimization variables of the nonlinear programs (NLPs), and the achieved average computation times, solving the model predictive controllers' underlying optimal control problems (OCPs) listed for the studied example systems.

		five-bar linkage		delta robot		manipulator with flexible link	
		gen.	abs.	gen.	abs.	min.	abs.
coordinates							
$n_q = \dim(\mathbf{q})$		4	12	9	30	4	10
$n_c = \dim(\mathbf{c})$		2	10	6	27	0	6
shooting intervals K		20	20	20	20	20	20
prediction horizon T		1 s	1 s	1 s	1 s	0.1 s	0.1 s
NLP variables	index-1 (impl. Euler)	208	544	438	1320	228	600
	index-1 (IDAS)						
	index-2 (expl. Euler)	288	944	678	2400	–	720
	index-3 (expl. Euler)	248	744	558	1860	–	600
	minimal (ODE)	–	–	–	–	228	–
avg. OCP computation time / ms	index-1 (impl. Euler)	77	67	1344	2586	unstable	
	index-1 (IDAS)	1871	1382	> 7 s / iteration		–	–
	index-2 (expl. Euler)	8	18	27	122	–	15
	index-3 (expl. Euler)	6	16	16	45	–	17
	minimal (ODE)	–	–	–	–	17	–

redundant description, presuming that mostly for systems where one can obtain a constraint-free description, a minimal (generalized-coordinate) approach would show advantages. However, the depicted results for the three implemented OCP variants of the delta robot prove such presumptions wrong. This can be attributed to the fact that the number of variables in the nonlinear program scales linearly with the number of coordinates, inputs, and constraints, so that the choice of MBS formulation directly reflects itself in the number of NLP variables. As such, when purely counting the number of optimization variables and constraints in the resulting OCPs, minimal formulations seem advantageous. However, conversely, as it is well-known in multibody literature, minimal formulations of MBS relevant in practical applications usually lead to highly nonlinear, interleaved terms in the equations of motion. Thus, many of the model-related expressions evaluated in each iteration of the employed numerical optimization algorithm become more expensive to evaluate, and appearing matrix expressions have less structure and sparsity that could be exploited by tailored numerical linear algebra algorithms. This makes for an interesting size-to-complexity trade-off. As the results show, in the studied cases, the larger size of OCPs based on redundant models seem to largely outweigh the benefits of simpler model expressions, even when using a modern optimization algorithm like IPOPT that is specifically designed for large-scale and sparse nonlinear programs.

To better understand in a quantitative manner the model-complexity baseline for this trade-off, Table 2 lists the number of processor instructions needed for the evaluation of the generalized forces as well as the generalized accelerations, i.e., $\ddot{\mathbf{q}} = \mathbf{M}^{-1}(\mathbf{h} + \mathbf{C}^\top \boldsymbol{\lambda})$, whereby the matrix inversion is performed symbolically in CasADi. Generally, the findings confirm the expectations with sometimes quite large differences to the benefit of redundant formulations. It is, however, worth to revisit the manipulator with flexible link. As can be confirmed from the recorded number of instructions, even for this simple system with four degrees of freedom, already the evaluation of the equations of motion is comparatively expensive, which is due to an outsize complexity increase due to the inclusion of the flexible body. Indeed, for both coordinate choices, the introduction of just one flexible body with one mode increases the effort by more than a factor of two when compared to a purely rigid implementation of the manipulator.

In any case, all things considered, the OCP formulations set forward in Sections 2.2, and 2.3 result in robust and fast MPC implementations that are in many cases already real-time capable, even though the workflow is still using the highly flexible and quite generally applicable CasADi and IPOPT through the Matlab interface. Further, it can be summarized that, in the studied scenarios, it never decisively pays off to use a redundant description and sometimes can be notably slower, particularly also in the presence of closed kinematic loops. However, the multibody-to-MPC toolchain still offers additional room for improvements without resorting to manual optimizations or tuning. We demonstrate this in the following using two additional measures, namely using an NLP solver that exploits characteristic linear-algebraic structures arising from MPC problems, and getting rid of overheads as well as speeding up evaluation by building the NLP directly in C/C++ and compiling with automatic code optimization.

3.3 Example: Structure-exploiting OCP Solver

The real-time application of model predictive control requires efficient NLP solvers to be able to reliably solve one OCP per sampling instance, which also enables to compensate for the delay incurred by the cal-

Table 2: Comparison of the complexity of the equations of motion, modeling the example systems using either absolute or minimal coordinates. The number of processor instructions read (ir) is recorded when evaluating the equations of motion in optimized C code, as recorded using the tool `callgrind` [41].

	five-bar linkage		delta robot		manipulator with flexible link	
coordinates	gen.	abs.	gen.	abs.	min.	abs.
$n_q = \dim(\mathbf{q})$	4	12	9	30	4	10
$n_c = \dim(\mathbf{c})$	2	10	6	27	0	6
gen. forces $\mathbf{h}$	382 ir	22 ir	1549 ir	510 ir	1256 ir	150 ir
gen. accelerations $\ddot{\mathbf{q}}$	1017 ir	540 ir	2518 ir	1605 ir	1930 ir	704 ir

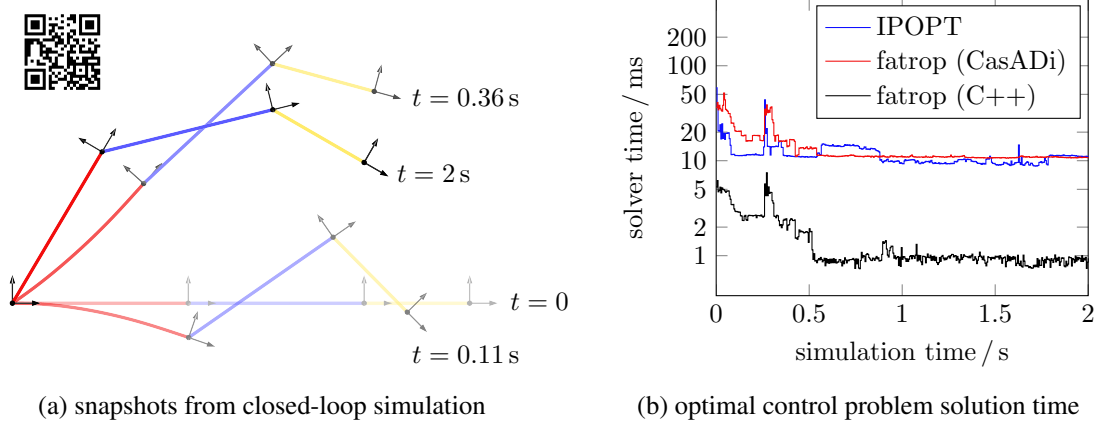


Figure 3: Simulation results with the flexible serial manipulator. Snapshots of the manipulator at selected times of the closed-loop trajectory are shown on the left. An animation [22] is accessible via the QR code. Two optimal control problem solvers, namely IPOPT and fatrop, are compared on the right.

culation time [11, Sec. 4.5.2]. Some of these solvers rely on sequential quadratic programming (SQP) and make a particular effort to benefit as much as possible from hotstarting the solution process using information from the previous time step, which is the case for real-time iteration schemes [6], e.g., implemented in acados [38]. Others rely on interior point methods, where specialized solvers exploit sparsity and block structures in matrices appearing in the solution process specifically for MPC-like problems, which is the case for fatrop [37]. For the latter, it is necessary to sort and structure variables and constraints in a certain manner, so that the matrix (sub-)structures appear as assumed in the optimization algorithm’s implementation. Otherwise, it is similarly flexible regarding the application to MPC problems as IPOPT and, thus, we subsequently study the application of fatrop.

First results are obtained for the delta robot, and at first again by accessing fatrop through CasADi’s Matlab interface. For the OCP formulations in redundant coordinates, a speed-up of up to 60 % is recorded, whereas OCP formulations in generalized coordinates still achieve a speed-up of up to 20 %. The difference in relative speed-up suggests that additional structure, i.e., sparsity connected to redundant multibody formulations, seems to be effectively exploited by the fatrop solver.

Next, the serial manipulator with flexible link is implemented and tasked to perform a set-point change for the pose of its end effector. Hereby, the control, i.e., the cost function, focuses on the end effector’s pose and velocity with a large penalty on deviations from the set-point, a small penalty on the elastic coordinate as well as its derivative, and finally a marginal penalty on the rigid links’ velocities. The latter penalty includes the floating reference frame as well and is mainly added for the purpose of better numerical conditioning. The resulting motion of the manipulator and the recorded computation times are displayed in Figure 3, which also includes a video [22] accessible through the provided quick response code. Both a formulation in minimal coordinates and the index-3 DAE formulation are implemented in an MPC loop and compared. However, as it is shown for the case of the minimal formulation in Figure 3b, the previously observed speed-up, calling fatrop instead of IPOPT from Matlab, is not directly achieved for this system. The same holds for the redundant index-3 DAE formulation, i.e., only marginal speed-up is achieved, when replacing IPOPT with fatrop.

Hence, a further investigation is carried out by isolating the optimal control problem from CasADi and Matlab by directly implementing it in C++, using the optimal control problem template provided by fatrop. Necessary gradients and the Hessian matrix of the OCP’s associated Lagrangian are computed and exported with CasADi. The fatrop library itself as well as the OCP source code are compiled with the GNU Compiler Collection [12] and optimization level $-O3$. The solver is then invoked directly from C++ and communicates with the simulation of the flexible manipulator in Matlab via the user datagram protocol (UDP). As it is depicted in Figure 3b using minimal coordinates and re-implementing the NLP in plain C++, computation times in the low millisecond range can be achieved, which is a speed-up of approximately one order of magnitude. This way, the results are very promising for real-time implementation, and quite remarkable for

the model predictive control of a flexible multibody system where the whole modeling and control design process works programmatically and fully automatically, without noteworthy hand-tuned or manually optimized program-code portions. Most likely the observed speed-up is connected to compiler optimization of the inevitably lengthy terms in the equations of motion that are introduced by the flexible body in floating frame of reference formulation. It is worth noting that CasADi also offers just-in-time (JIT) compilation options that make compiler optimizations available without fully switching to a C++ implementation, although we, here, opted for a clean C++ implementation to avoid any residual effects of interfacing between Matlab and C++.

Beyond quantitative observations, there also appear interesting conceptual differences between different formulations that can be relevant for application practice. In that regard, one such advantage of using higher-index DAE formulations in the OCP is demonstrated subsequently.

3.4 Example: Constraining Joint Forces using the Higher-index DAE Formulations

In the approaches set forward in Sections 2.2 and 2.3, the Lagrange multipliers λ_k are treated as optimization variables, allowing to impose penalties, limits, or other constraints on them. Naturally, the Lagrange multipliers are not fully independent variables, but they can be influenced through the executed motion which, in turn, is influenced by the freely manipulable control inputs. Mechanically, limits on Lagrange multipliers result in corresponding limits on the reaction forces. Generally, a reaction force is a linear combination of multiple Lagrange multipliers with the relation given by the Jacobian matrix C . In our case, the constraint equations are formulated in the fixed inertial frame, such that the reaction force components of interest directly correspond to entries in the vector λ . Multibody system formulations in redundant coordinates are more directly useful for limiting reaction forces, since all joints are usually represented by kinematic constraints with corresponding multipliers. Using minimal coordinate formulations, only the loop-closing constraints are directly formulated as kinematic constraints, and recovering reaction forces in other joints and hinges does require additional computations. However, even when using redundant coordinates, when using NLP formulations with prepackaged implicit integration schemes, the Lagrange multipliers are capsuled inside the integration scheme and are not directly available as optimization variables. With such a setup, the subsequent example would be impossible to realize and, instead, separate, careful trajectory planning can be an option, although an often less elegant and more conservative one.

As an example, the previously introduced five-bar linkage is used, performing a point to point motion of the end effector, as it is shown in Figure 4. In the original configuration, the optimal control problem is set up to minimize the squared distance between the end effector's position and the target point, respecting the system dynamics and simple bounds on the control input u . The modified OCP is additionally equipped with the constraint $\|\lambda_{J_1,k}\|^2 \leq \bar{\lambda}, k \in \{0, 1, \dots, N-1\}$, whereby λ_{J_1} corresponds to the two Lagrange multipliers that enforce the constraints implied by the joint and $\bar{\lambda}$ represents an empirically chosen upper limit. Figure 4 shows the results of the two closed-loop simulations. As can be seen, the initial peak in the evolution of the reaction force is effectively suppressed, which the optimization algorithm achieves by, effectively, executing a slower motion than in the unconstrained-multiplier case. Additionally to the slower speed and quite interestingly, the resulting end-effector trajectory is no longer a direct connection between start point and target, which, however, was also not stated as a control goal, and is hence a valid choice if it helps to reduce the reaction force. This could be a simple and elegant approach in many applications, such as in pick-and-place tasks, providing an effective way to adhere to hardware limitations without the need of precomputed trajectories, and potentially helping to reduce material fatigue and wear and tear of bearings and components. In a similarly simple manner, instead of constraining reaction forces, one can penalize their absolute values in the cost function, leading to a softer, configurable compromise between the original control goals and the added reaction-force criterion than offered by hard constraints. This can also prevent issues with feasibility if one tries to impose overly restrictive constraints on the reaction forces that cannot be fulfilled just by influencing the control inputs.

3.5 Remarks on Consistent Initial Conditions

Before concluding the article, it is important to give some remarks about consistent initial conditions, which is a highly relevant topic when using different simulation models for control and closed-loop simulation or

when dealing with real-world measurements. All OCP formulations discussed in this contribution require the initialization of the OCP, i.e., the provided system state, to be consistent with the kinematic constraints. In the case of NLP (5), this requirement is inferred from the implicit integration scheme, for which convergence guarantees always assume consistent initial conditions [19]. In the case of NLPs (6) and (10) the assumption is made as well to obtain meaningful predicted trajectories. However, especially in practice, a full state measurement is hardly ever available to the controller, such that additional effort to obtain a consistent state estimation becomes necessary.

Regarding rigid multibody systems, the coordinate partitioning approach [23] comes back to mind, measuring a subset of coordinates and computing all dependent coordinates before feeding the state back into the OCP. To obtain the dependent positions, a system of nonlinear equations must be solved, whereas the dependent velocities follow from the solution of a linear equation system. For elastic MBS, the situation is even more complex, since the control model will usually not be able to cover every elastic deformation for reasons of efficiency. Already in simulation, as observed with the flexible manipulator, using more eigenmodes in the simulation model than present in the control model leads to unavoidable constraint violations, despite the full state information being available. In a real-world experiment, this situation generally would only become worse. Hence, the arguably most natural solution is to interpret the necessity of obtaining constraint-fulfilling initial states as a constrained estimation problem and treating it jointly with the, anyways often required, state estimator. As such, common variants of the Kalman filter, which represent the by far most common choices for (optimal) state estimators, are not particularly attractive since they do not offer a direct way of dealing with constraints. Instead, moving horizon estimation (MHE) is of particular interest in this context. As MHE is the dual of MPC, for optimal estimation instead of for control, similar considerations like in the present article are relevant, as detailed in upcoming work. Consequently, when being determined to use presently more common and popular state estimation approaches, minimal models are of clear conceptual advantage as, for simple systems, they will not have any constraints, permitting direct usage of typical Kalman filter variants, or will at least have as few constraints as possible, meaning fewer constraints would need to be dealt with by hand-crafted auxiliary approaches.

4 DISCUSSION: PERSPECTIVES FOR MULTIBODY MODELING SOFTWARE

The article's main contribution and findings allow some remarks and observations on beneficial qualities that multibody modeling software should possess to be fit for real-time optimal control like MPC. As such, in order to efficiently incorporate multibody models into fast optimization-based controllers, the multibody-modeling software is required to provide evaluable functions for the equations of motion, constraint equations, associated gradients, and further quantities, together with an adequate documentation. However, depending on the optimization algorithm or the optimization problem to be solved, different requirements arise such that a multibody code might need to expose a large variety of said functions, which all have

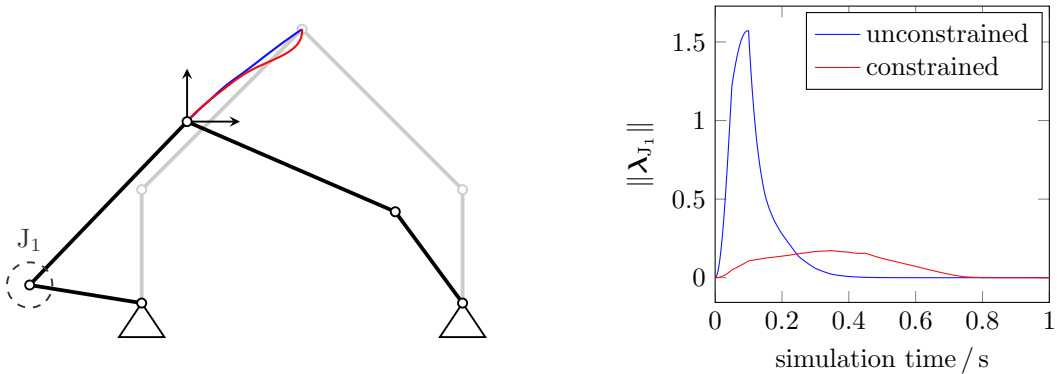


Figure 4: Simulation results performing a point to point motion with the rigid five-bar linkage, investigating the resulting reaction forces in the marked joint. The resulting trajectories with and without constraints on the magnitude of the reaction force within the joint are displayed on the left. On the right, the corresponding values of said reaction force are shown with respect to the simulation time.

to be adequately maintained, and which can become particularly cumbersome for expressions resulting from derivatives, like gradients and Hessian matrices. An elegant solution to this whole issue is a symbolic representation of selected, individual terms of the equations of motion, as they are used within this publication. All subsequent manipulations, e.g., for formulating OCPs, may then be performed with the same symbolic engine or, if compatible, even with another symbolic engine tailored to OCP needs, to arrive at the individually needed ingredients for a discretized optimal control problem. As an alternative, if the multibody-modeling software provides code for numerical evaluation only, when appropriately formulated, algorithmic differentiation techniques may still be used to derive the necessary gradients, etc., but we suspect this approach to often be less efficient, as it, e.g., provides less potential for optimizations of formulations and code than a symbolic formulation. Further, symbolic frameworks may more easily support the joint design of control and system, e.g., to jointly optimize system and controller design for optimal performance, helping to overcome the unnatural but often perceived barrier between dynamics and control. Further, the article’s results show that minimal (generalized-coordinate) approaches quite regularly lead to shorter computation times and hence more efficient optimal control problems, even in those cases where a minimal set of (kinematic-loop-closure) constraints remains. This suggests that multibody modeling software fit for real-time optimal control should not only offer formulations based on redundant coordinates when maximum efficiency is sought in each case. Still, when fast enough, in some cases, redundant-coordinate approaches can be preferable or simply easier to use, see, e.g., the example given in Section 3.4. Hence, this calls for maximum flexibility not only regarding export options but also regarding supported coordinate choices.

In any case, one cannot overstate the importance of the system dynamics’ availability in the form of standalone equations of motion that are not yet interconnected with a numerical integration scheme especially for DAE-based multibody models. This is quite often not satisfactorily implemented in software relying for model export mostly on the functional mockup interface (FMI) [36], especially in co-simulation mode. Indeed, as this article’s results have shown, for optimal-control purposes, the immediate packaging with a numerical integrator is hindering and sometimes even preventing the efficient solution of OCPs using DAE-based multibody models, see also the comments made in Section 2.1. In model-exchange mode, the FMI standard version 3 moves towards a broader class of systems that can be described. Still, to the authors’ knowledge, the handling of DAEs is not yet straight forward. On the positive side, FMI requires implementations of the underlying models or co-simulation containers in C/C++ code, which allows the deployment of the code to a wide range of devices, especially embedded control hardware, as, otherwise, the ability to export C/C++ code is not granted. However, this still means that a lot of multibody modeling software packages presently popular with users, e.g., for simulation workflows, indeed do not conveniently support those optimal control problem formulations identified as preferable in this article and cannot be directly used in an automated toolchain from model to efficient controller. In some respects, it even seems like the hardship to obtain MPC-fit models from popular modeling tools explains some popularity of reinforcement learning (RL) to solve optimal control problems that, in principle and at the current state of the art, could be solved more reliably by MPC, in particular in those cases where still a (multibody) simulator is used to train the controller (actor), as then, the simulation model only needs to be evaluated and not exported in the form of individual and differentiable components.

An automated workflow from well formulated multibody model to real-time-deployable MPC controller can also prove to be a valuable building block in a larger toolchain following from other contemporary developments. As such, large-language models (LLMs) can be used to interact with anyway often text-based interfaces of multibody modeling software to automate modeling and implementation processes further [16, 40], albeit without success guarantees, as is to be expected for today’s LLM-based approaches [21]. Combining a language-model-based design assistant for multibody models with an automated workflow from multibody model to MPC controller thereby is nowadays readily conceivable, furnishing a language-driven design assistant for controlled (smart) multibody systems, at least if the control goals are clear or also inferred from language.

5 CONCLUSION AND OUTLOOK

The work at hand set forward and intently analyzed automatable approaches to formulate optimal control problems for real-time feedback control from multibody system models in the form of higher-index

differential-algebraic equations. Despite such models being extremely common in multibody dynamics, an in-depth and comparative treatment of this situation for real-time optimal control was so far lacking. Rather than performing an index reduction and discretizing the resulting index-1 DAE with an implicit integration scheme, it is shown to be beneficial to use an explicit integration of the unconstrained system dynamics while treating the Lagrange multipliers as additional optimization variables and, thus, letting the kinematic constraints be taken care of by the optimization algorithm by including them in the discrete-time optimal control problem as position-level constraints. The approach is highly efficient with regard to computation time as it is exemplarily shown for three selected multibody systems. Further, the article sets forward and compares an otherwise similar approach that relies on index reduction to index 2, which can be implemented more easily for less flexible MPC solution algorithms. In addition, an in-depth comparative performance analysis, also including different coordinate choices, provides guidance on practical modeling choices in the light of MPC applications, highlighting the value of minimal-coordinate approaches even when some loop-closure constraints remain. In future research the assumption of a holonomic system could be relaxed to cover the often occurring nonholonomic dynamics in the field of robotics. Particularly, asymptotically stabilizing states of nonholonomic systems generally requires discontinuous feedback [3, 5], turning MPC's implicit-feedback setup actually into an advantage [30, 8], even more so if the underlying OCP can be solved robustly and effectively. Finally, this article has also emphasized that, for the model predictive control of DAE-based models, consistent initial conditions in the form of full state information in each sampling interval have to be available. Ongoing research investigates efficient realizations of moving horizon estimation (MHE) for this purpose.

REFERENCES

- [1] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. CasADi – a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.
- [2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, 2004.
- [3] R. W. Brockett. Asymptotic stability and feedback stabilization. In *Differential Geometric Control Theory*, volume 27, pages 181–191, Houghton, 1983.
- [4] M. Burkhardt, R. Seifried, and P. Eberhard. Aspects of symbolic formulations in flexible multibody systems. *Journal of Computational and Nonlinear Dynamics*, 9(4):041013–1–041013–8, 2014.
- [5] J.-M. Coron. A necessary condition for feedback stabilization. *Systems & Control Letters*, 14(3):227–232, 1990.
- [6] M. Diehl, H. Bock, H. Diedam, and P.-B. Wieber. *Fast direct multiple shooting algorithms for optimal robot control*, pages 65–93. *Fast Motions in Biomechanics and Robotics: Optimization and Feedback Control*. Springer, Heidelberg, 2006.
- [7] Q. Docquier, O. Brüls, and P. Fiset. Comparison and analysis of multibody dynamics formalisms for solving optimal control problem. In *IUTAM Symposium on Intelligent Multibody Systems – Dynamics, Control, Simulation*, pages 55–77, Sozopol, 2019.
- [8] H. Ebel, M. Rosenfelder, and P. Eberhard. A note on the predictive control of non-holonomic systems and underactuated vehicles in the presence of drift. *PAMM*, 23(4):e202300022, 2023.
- [9] T. Englert, A. Völz, F. Mesmer, S. Rhein, and K. Graichen. A software framework for embedded non-linear model predictive control using a gradient-based augmented Lagrangian approach (GRAMPC). *Optimization and Engineering*, 20(3):769–809, 2019.
- [10] J. Fehr, D. Grunert, P. Holzwarth, B. Fröhlich, N. Walker, and P. Eberhard. Morembs – a model order reduction package for elastic multibody systems and beyond. In *Reduced-Order Modeling (ROM) for Simulation and Optimization: Powerful Algorithms as Key Enablers for Scientific Computing*, pages 141–166. Springer International Publishing, Cham, 2018.

- [11] R. Findeisen. *Nonlinear Model Predictive Control: A Sampled-Data Feedback Perspective*. VDI, Düsseldorf, 2005.
- [12] Free Software Foundation. *GCC, the GNU compiler collection (version 14.2)*. GNU Project, 2024.
- [13] J. García de Jalón, J. Unda, and A. Avello. Natural coordinates for the computer analysis of multibody systems. *Computer Methods in Applied Mechanics and Engineering*, 56(3):309–327, 1986.
- [14] C. W. Gear, B. Leimkuhler, and G. K. Gupta. Automatic integration of Euler-Lagrange equations with constraints. *Journal of Computational and Applied Mathematics*, 12&13:77–90, 1985.
- [15] J. Gerstmayr. Exudyn – a C++ based python package for flexible multibody systems. *Multibody System Dynamics*, 60:533–561, 2024.
- [16] J. Gerstmayr, P. Manzl, and M. Pieber. Multibody models generated from natural language. *Multibody System Dynamics*, 62:249–271, 2024.
- [17] S. Gros, M. Zanon, M. Vukov, and M. Diehl. Nonlinear MPC and MHE for mechanical multi-body systems with application to fast tethered airplanes. *IFAC Proceedings Volumes*, 45(17):86–93, 2012. 4th IFAC Conference on Nonlinear Model Predictive Control.
- [18] L. Grüne and J. Pannek. *Nonlinear Model Predictive Control: Theory and Algorithms*. Springer, Cham, second edition, 2017.
- [19] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*. Springer, Berlin, 2 edition, 1996.
- [20] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):363–396, 2005.
- [21] A. T. Kalai, O. Nachum, S. S. Vempala, and E. Zhang. Why language models hallucinate. *Preprint arXiv:2509.04664*, 2025.
- [22] A. Kargl, H. Ebel, and P. Eberhard. Supplementary material from investigating advantageous formulations of multibody systems for optimal control. 2026. <https://doi.org/10.18419/DARUS-5751>.
- [23] T. Kurz, M. Burkhardt, and P. Eberhard. Systems with constraint equations in the symbolic multibody simulation software Neweul-M². In *Proceedings of the ECCOMAS Thematic Conference on Multibody Dynamics 2011*, Brussels, Belgium, 2011.
- [24] T. Kurz, P. Eberhard, C. Henninger, and W. Schiehlen. From Neweul to Neweul-M²: Symbolical equations of motion for multibody system analysis and synthesis. *Multibody System Dynamics*, 24(1):25–41, 2010.
- [25] G. Leister, E. Kreuzer, and W. Schiehlen. The software NEWEUL. In W. Kortüm and R. S. Sharp, editors, *Multibody Computer Codes in Vehicle System Dynamics*, pages 95–98, Amsterdam, 1993. Swets & Zeitlinger. ISBN 9-026-51365-8.
- [26] K. Magnus and H. Müller. *Grundlagen der Technischen Mechanik*. Teubner, Stuttgart, 1990.
- [27] C. Pöll and I. Hafner. Index reduction and regularisation methods for multibody systems. *IFAC-PapersOnLine*, 48(1):306–311, 2015.
- [28] R. Quirynen, M. Vukov, M. Zanon, and M. Diehl. Auto-generating microsecond solvers for nonlinear MPC: A tutorial using ACADO integrators. *Optimal Control Applications and Methods*, 36(5):685–704, 2014.
- [29] J. Rawlings, D. Q. Mayne, and M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, Santa Barbara, 2nd edition, 2017.
- [30] M. Rosenfelder, H. Ebel, J. Krauspenhaar, and P. Eberhard. Model predictive control of non-holonomic systems: Beyond differential-drive vehicles. *Automatica*, 152:110972, 2023.

- [31] T. Samad, M. Bauer, S. Bortoff, S. Di Cairano, L. Fagiano, P. F. Odgaard, R. R. Rhinehart, R. S.-P. na, A. Serbezov, F. Ankersen, P. Goupil, B. Grosman, M. Heertjes, I. Mareels, and R. Sasseh. Industry engagement with control research: Perspective and messages. *Annual Reviews in Control*, 49:1–14, 2020.
- [32] W. Schiehlen and P. Eberhard. *Applied Dynamics*. Springer, Cham, 2014.
- [33] S. Schneider and P. Betsch. Optimal control of constrained mechanical systems in redundant coordinates: Formulation and structure-preserving discretization. *Computer Methods in Applied Mechanics and Engineering*, 432:117443, 2024.
- [34] R. Schwertassek and O. Wallrapp. *Dynamik flexibler Mehrkörpersysteme*. Vieweg, Braunschweig, 1999.
- [35] A. Tasora, R. Serban, H. Mazhar, A. Pazouki, D. Melanz, J. Fleischmann, M. Taylor, H. Sugiyama, and D. Negrut. Chrono: An open source multi-physics dynamics engine. In *High Performance Computing in Science and Engineering*, pages 19–49. Springer, 2016.
- [36] The Modelica Association Project FMI. Functional mock-up interface specification. 2024. Available online: <https://fmi-standard.org>.
- [37] L. Vanroye, A. Sathya, J. De Schutter, and W. Decré. Fatrop: A fast constrained optimal control problem solver for robot trajectory optimization and control. In *Proceedings of the 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 10036–10043, Detroit, 2023.
- [38] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl. acados – a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*, 14(1):147–183, 2022.
- [39] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.
- [40] J. Wang, H. Zhang, K. Slaton, S. Wang, R. Serban, J. Wu, and D. Negrut. ChronoLLM: A framework for customizing large language model for digital twins generalization based on PyChrono. *Preprint arXiv:2501.04062*, 2025.
- [41] J. Weidendorfer, M. Kowarschik, and C. Trinitis. *A tool suite for simulation based analysis of memory access behavior*, pages 440–447. Springer Berlin Heidelberg, 2004.
- [42] C. Woernle. *Multibody Systems*. Springer, Berlin, 2024.