The 5th Joint International Conference on Multibody System Dynamics
June 24 – 28, 2018, Lisboa, Portugal

Analysis of Servo-constraints Solution Approaches for Underactuated Multibody Systems

Svenja Otto[1] and Robert Seifried[1]

[1] *Institute of Mechanics and Ocean Engineering, Hamburg University of Technology, {svenja.otto,robert.seifried}@tuhh.de*

ABSTRACT — *The servo-constraints approach is an efficient method for computing inverse models of underactuated multibody systems. Underactuated multibody systems possess more degrees of freedom than independent control inputs. The inverse model can be used as a feedforward controller in a two degree of freedom control structure. Servo-constraints constrain the output to a specified trajectory and append the equations of motion to form a set of differential-algebraic equations (DAEs). The resulting DAEs might be of higher differentiation index and are thus difficult to solve numerically. Here, different solution and analysis methods for the servo-constraints approach are compared. Especially, various solvers are analyzed with respect to real-time capability and accuracy.*

1 Introduction

Trajectory tracking of multibody systems is usually pursued in terms of a two degree of freedom control structure. Ideally, the feedforward controller is designed as an inverse model. It provides system inputs that exactly maneuver the system on a specified trajectory if there are no disturbances or modeling errors. The feedforward path is appended by a state feedback path to account for possible disturbances or modeling errors and stabilize the motion around the specified trajectory. This control structure is depicted in Fig. 1. With an accurate inverse model, the tracking errors are usually small and simple state feedback strategies such as a linear quadratic regulator are sufficient for disturbance rejection.

The described control structure is also desirable for underactuated multibody systems. For that purpose an accurate inverse model is required. Underactuated multibody systems possess more degrees of freedom than independent control inputs. Due to underactuation, it is not straightforward to derive an inverse model for these systems. For example, the Byrnes/Isidori input-output normal form approach can be applied for systems in minimal coordinates [1] or systems in redundant coordinates [2]. However, even for small and simple systems, the equations tend to become complex and difficult to analyze.

Instead of deriving the inverse model analytically, the servo-constraints approach computes the inverse model numerically [3]. The equations of motion are appended by servo-constraints which constrain the output to stay on a specified trajectory. Thereby, the equations of motion can be either formulated in minimal or redundant coordinates, see [4]. The resulting differential-algebraic equations (DAEs) representing the inverse model are often of higher differentiation index [5]. The solution of the DAEs includes the desired control input which moves

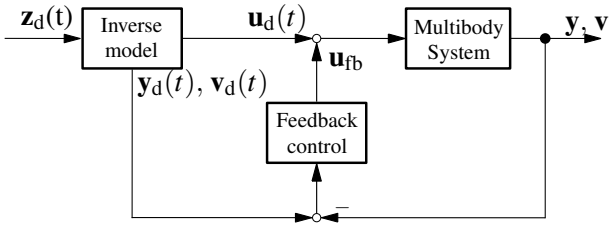


Fig. 1: Two degree of freedom control structure.

the multibody system on the specified trajectory. For application, the desired control input is ideally calculated in real-time. Then, it is possible to change the trajectory or even the system model online.

Most commonly, the implicit Euler scheme is used in the framework of servo-constraints to solve the DAEs [3], [4], [6]. Its advantages are simple implementation, real-time capability and stability properties [7]. However, the implicit Euler is known for numerical damping issues [8].

So far, the servo-constraints approach is mostly applied to differentially flat systems, e.g. in [3]. Differentially flatness is introduced in [9] and describes systems for which the system input can be written as a function of the output and a finite number of its derivatives. Since the inverse model can thus be derived analytically, it is used as reference for the numerical solution obtained from the DAEs. In case of differential flatness, the inverse model itself is an algebraic model. Thus, it does not exhibit any dynamics and damping of the implicit Euler does not reduce the solution accuracy.

However, servo-constraints can also be applied to systems with a non-flat output [6]. In that case, internal dynamics remain during model inversion. Internal dynamics is a concept from nonlinear control theory, see e.g. [1] and describes dynamics which cannot be observed from the input-output behavior of a system. In that case, the implicit Euler damps out the dynamics and might not be applicable to solve the inverse model DAEs for such systems. First approaches for applying servo-constraints to systems with internal dynamics are for example presented in [6] for stable internal dynamics. In order to deal with unstable internal dynamics, the feedforward control problem can be formulated as a boundary value or as an optimization problem. Both approaches are compared in [10].

Most DAE solvers are developed for DAEs with differentiation index 1. In case the differentiation index is larger than one, the problem is ill-conditioned [7]. Therefore, index reduction methods are common to simplify the numerical solution process. Common choices in the context of servo-constraints are minimal extension [11] and projection [3]. A reformulation as an optimization problem is proposed in [12] in order avoid numerical issues arising with solving DAEs directly. However, real-time capability cannot be assured in this approach.

In the following, the servo-constraints approach is reviewed in Section 2 and analysis as well as solution approaches are compared. The focus is on a selection of DAE solvers including implicit Runge Kutta as well as BDF methods. The solvers are compared in Section 3 with respect to accuracy, damping and real-time applicability. The results are stated for an overhead crane as a representative example for differentially flat systems and for a mass-on-car system as a representative example of a system with non-flat output. Experimental results support the results for the overhead crane. The results are summarized in Section 4.

2 Analysis of Servo-constraints

Holonomic underactuated multibody systems without kinematic loops are considered. They can be modeled using either minimal or redundant coordinates. The minimal position coordinates $\mathbf{y} \in \mathbb{R}^n$ and velocity coordinates $\mathbf{v} \in \mathbb{R}^n$ are chosen for a system with n degrees of freedom. Let the system input be $\mathbf{u} \in \mathbb{R}^{n_\mathrm{u}}$, where $n_\mathrm{u} < n$ is the number of independent system inputs. Applying the Newton Euler formalism yields a set of $2n$ differential equations of the form

$$\dot{\mathbf{y}} = \mathbf{Z}(\mathbf{y})\,\mathbf{v} \tag{1}$$

$$\mathbf{M}(\mathbf{y},t)\,\dot{\mathbf{v}} = \mathbf{q}(\mathbf{y},\mathbf{v},t) + \mathbf{B}\,\mathbf{u}\,. \tag{2}$$

Thereby, $\mathbf{Z} : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ is the kinematics matrix, $\mathbf{M} : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^{n\times n}$ denotes the mass matrix, $\mathbf{q} : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ describes the forces acting on the system and $\mathbf{B} \in \mathbb{R}^{n\times n_\mathrm{u}}$ is the input distribution matrix. Note that in two dimensional problems with minimal coordinates, it is mostly $\mathbf{Z} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. The system output $\mathbf{z} \in \mathbb{R}^{n_\mathrm{z}}$ is a function of the position coordinates

$$\mathbf{z} = \mathbf{h}(\mathbf{y}) \tag{3}$$

and it is assumed that the number of inputs equals the number of outputs, $n_\mathrm{u} = n_\mathrm{z}$. In case of kinematic loops, a formulation of the dynamics in form of Eqs. (1)-(2) might not always be possible. Moreover, it is sometimes helpful to rewrite the dynamics using redundant coordinates. The redundant position coordinates $\mathbf{y} \in \mathbb{R}^{n+n_\mathrm{c}}$ and velocity coordinates $\mathbf{v} \in \mathbb{R}^{n+n_\mathrm{c}}$, where n_c is the number of constraints, are chosen. Note that in a p-body system, the number of coordinates is $n+n_\mathrm{c} = 6p$, if all bodies are described using redundant coordinates. The equations of motion in DAE form arise as

$$\dot{\mathbf{y}} = \mathbf{Z}(\mathbf{y})\,\mathbf{v} \tag{4}$$

$$\mathbf{M}(\mathbf{y},t)\,\dot{\mathbf{v}} = \mathbf{q}(\mathbf{y},\mathbf{v},t) + \mathbf{C}_\mathrm{g}^\mathrm{T}(\mathbf{y})\,\boldsymbol{\lambda} + \mathbf{B}\,\mathbf{u} \tag{5}$$

$$\mathbf{c}_\mathrm{g}(\mathbf{y}) = \mathbf{0}\,, \tag{6}$$

with matrices of respective size for the $n+n_\mathrm{c}$ coordinates. The constraints on position level are denoted by $\mathbf{c}_\mathrm{g} : \mathbb{R}^{n+n_\mathrm{c}} \to \mathbb{R}^{n_\mathrm{c}}$. The generalized reaction forces are denoted by $\boldsymbol{\lambda} \in \mathbb{R}^{n_\mathrm{c}}$ and they are distributed onto the coordinate directions by the constraint gradient matrix $\mathbf{C}_\mathrm{g} : \mathbb{R}^{n+n_\mathrm{c}} \to \mathbb{R}^{n_\mathrm{c}\times n+n_\mathrm{c}}$. Since the inclusion of servo-constraints will yield a DAE system anyways, both formulations of Eqs. (1)-(2) as well as (4)-(6) can be used in the approach presented in the following. Therefore, the formulations will be treated uniformly and a distinction is only noted if necessary.

2.1 Problem Statement

For trajectory tracking of underactuated multibody systems, feedforward control based on accurate inverse models is a convenient control method. In order to obtain an inverse model of the underactuated multibody system either in form of Eqs. (1)-(2) or in form of Eqs. (4)-(6), the servo-constraints approach is used here. This yields a numerical representation of the inverse model.

The servo-constraints $\mathbf{c} : \mathbb{R}^{n+n_\mathrm{c}} \times \mathbb{R} \to \mathbb{R}^{n_\mathrm{u}}$ enforce the output $\mathbf{z}$ to follow a specified output trajectory $\mathbf{z}_\mathrm{d}(t)$. They append the model dynamics to form a set of differential-algebraic equations

$$\dot{\mathbf{y}} = \mathbf{Z}(\mathbf{y})\,\mathbf{v} \tag{7}$$

$$\mathbf{M}(\mathbf{y},t)\,\dot{\mathbf{v}} = \mathbf{q}(\mathbf{y},\mathbf{v},t) + \mathbf{C}_\mathrm{g}^\mathrm{T}(\mathbf{y})\,\boldsymbol{\lambda} + \mathbf{B}\,\mathbf{u} \tag{8}$$

$$\mathbf{c}_\mathrm{g}(\mathbf{y}) = \mathbf{0} \tag{9}$$

$$\mathbf{c}(\mathbf{y},t) = \mathbf{z}(\mathbf{y}) - \mathbf{z}_\mathrm{d}(t) = \mathbf{0}\,. \tag{10}$$

These $2n+n_\mathrm{u}+3n_\mathrm{c}$ equations are solved for the unknown position and velocity coordinates $\mathbf{y}_\mathrm{d}$ and $\mathbf{v}_\mathrm{d}$, the respective generalized reaction forces $\boldsymbol{\lambda}_\mathrm{d}$ and the desired control input $\mathbf{u}_\mathrm{d}$ that keeps the system on the desired output trajectory $\mathbf{z}_\mathrm{d}(t)$. The desired input $\boldsymbol{u}_\mathrm{d}$ is used as feedforward control and the desired state trajectories $\mathbf{y}_\mathrm{d}$ and $\mathbf{v}_\mathrm{d}$ are used as reference for any state feedback controller in the control structure shown in Fig. 1. In case of using minimal coordinates, the generalized reaction forces $\boldsymbol{\lambda}$ and respective constraints $\mathbf{c}_\mathrm{g}$ vanish.

The differential-algebraic Eqs. (7)-(10) representing the inverse model have a comparable structure to the multibody dynamics in DAE form of Eqs. (4)-(6). While the geometric constraints $\mathbf{c}_\mathrm{g}$ are enforced by the generalized reaction forces $\boldsymbol{\lambda}$, the servo-constraints $\mathbf{c}$ are enforced by the system input $\mathbf{u}$. In contrast to the reaction forces, the system inputs are not necessarily orthogonal to the tangent of the respective constraint manifold [3]. Thus, the arising differentiation index of the inverse model DAEs might be larger than 3. Roughly speaking, the differentiation index describes the minimal number of differentiations of the algebraic equations which are necessary to obtain an ordinary differential equation (ODE) for the algebraic variables. See e.g. [13] for a definition and classification of the differentiation index. Also, the dynamics of the inverse model might be much more complex than the forward dynamics.

The solutions can be classified into different configurations. In case of differentially flat systems, the inverse model is purely algebraic. These systems are exactly linearizable by a coordinate transformation and dynamic or quasi-static state feedback [14]. For single-input-single-output (SISO) systems, state feedback linearization is possible in case the relative degree r of a system is equal to the number of states in state space, $r = 2n$ for

holonomic systems formulated in minimal coordinates. The relative degree is defined as the number of times the system output must be differentiated for the system input to appear for the first time. This result can be generalized to a vector relative degree for multi-input-multi-output (MIMO) systems [15]. In case of systems with non-flat output, the inverse model itself is a dynamic system. For holonomic SISO systems, internal dynamics of dimension $2n-r$ remain. This result can be generalized to MIMO systems as well. In order to apply the inverse model as feedforward control, the internal dynamics have to be stable.

2.2 Analysis Approaches

Internal dynamics describes the dynamics which remains unobservable from the input-output relationship and is a concept from nonlinear control theory [1]. In order to explicitly obtain a formulation of the internal dynamics, it is convenient to perform a coordinate transformation using the new coordinates

$$\tilde{\mathbf{y}} = \begin{bmatrix} \mathbf{z} \\ \mathbf{y}_\mathrm{u} \end{bmatrix} = \begin{bmatrix} \mathbf{h}(\mathbf{y}) \\ \mathbf{y}_\mathrm{u} \end{bmatrix}, \tag{11}$$

where $\mathbf{y}_\mathrm{u}$ denote the unactuated coordinates. See [6] for detailed derivation of the internal dynamics.

In order to directly apply the system inversion based on servo-constraints, the internal dynamics must be stable, which is equivalent to the notion of a minimum-phase system. Then, the solution of Eqs. (7)-(10) yields bounded desired inputs $\mathbf{u}_\mathrm{d}$. The internal dynamics is usually nonlinear and driven by the output trajectory $\mathbf{z}$. It is therefore difficult to analyze it with respect to stability. Setting the output and its derivatives to zero $\mathbf{z}(t) = \dot{\mathbf{z}}(t) = \ddot{\mathbf{z}}(t) = \mathbf{0}\ \forall t$ yields the zero dynamics [1]. Its stability is usually analyzed by Lyapunov's indirect method. Local exponential stability of the zero dynamics guarantees stability of the internal dynamics if the desired output trajectory $\mathbf{z}_\mathrm{d}(t)$ and its first $r-1$ derivatives are small in magnitude [15].

2.3 Numerical Solution Approaches

The higher index DAEs (7)-(10) describing the inverse model must be solved preferably in real-time for the desired control input $\mathbf{u}_\mathrm{d}$. Note that in general, a solution of nonlinear DAEs cannot be guaranteed. Here, the initial conditions $\mathbf{y}_0$, $\mathbf{v}_0$, $\boldsymbol{\lambda}_0$ and $\mathbf{u}_0$ are assumed to be consistent with the constraint Eqs. (9)-(10). Consistent initial conditions can be found e.g. by solving for the equilibrium of the multibody system. Moreover, the servo-constraint Eqs. (10) are assumed to be compatible with possible motion of the multibody system.

A suitable DAE solver must be carefully chosen and the solution should be monitored carefully. This is for example possible by applying the calculated control input in a forward-time integration and comparing the simulated system output with the desired trajectory. An overview of suitable DAE solvers is e.g. given in [7], [16]. Since higher index DAEs are usually ill-conditioned, several approaches reduce the index. In the context of servo-constraints, minimal extension [11] or projection onto the constrained and unconstrained manifold [3] are common options.

Here, the focus is on comparing the solver class of implicit Runge-Kutta methods and backwards differencing formulas. Both can be applied to either the original high index formulation or a reduced index formulation. Let the DAE system which is to be solved be summarized in the implicit form

$$\mathbf{F}(\mathbf{x}, \dot{\mathbf{x}}, t) = \mathbf{0}, \tag{12}$$

where the Jacobian $\frac{\partial \mathbf{F}}{\partial \dot{\mathbf{x}}}$ is singular and $\mathbf{x}$ contains all unknown variables $\mathbf{y}$, $\mathbf{v}$, $\boldsymbol{\lambda}$ and $\mathbf{u}$.

Implicit Runge-Kutta methods

Implicit Runge-Kutta schemes are single-step integration schemes, for which the solution $\mathbf{x}_n$ at time t_n depends on the solution at the last time step t_{n-1} and a number of s function evaluations in between. Here, constant time step integrations with step size $h = t_n - t_{n-1} = const.$ are considered. Thus, they can be implemented on a real technical

setup running with constant frequency. Following [16], the s-stage scheme for DAEs in implicit form of Eq. (12) is given by

$$\mathbf{F}\left(\mathbf{x}_{n-1} + h\sum_{j=1}^{s} a_{ij}\mathbf{X}'_j, \mathbf{X}'_j, t_{n-1} + c_i h\right) = \mathbf{0} \qquad i = 1,2,...,s \tag{13}$$

$$\mathbf{x}_n = \mathbf{x}_{n-1} + h\sum_{i=1}^{s} b_i \mathbf{X}'_j\,. \tag{14}$$

The derivatives at the time steps $t_{n-1} + c_i h$ in between t_n and t_{n-1} are denoted by $\mathbf{X}'_j$. These values are obtained by solving the nonlinear Eqs. (13) of dimension $s \cdot N$, where N is the size of the original Eq. (12) which is to be solved. The coefficients a_{ij}, b_i and c_i define the convergence order of the method and are usually collected in the matrices $\mathbf{A}$, $\mathbf{b}$ and $\mathbf{c}$ written in the butcher tableau

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^{\mathrm{T}} \end{array}. \tag{15}$$

The implicit Euler scheme with $s = 1$ stage is usually applied in the context of servo-constraints. Here it is compared to the 3-stage Radau IIA scheme of order 5, denoted by Radau5. Both parameter sets are summarized in Tab. 1. In both schemes, the nonlinear Eqs. (13) are solved with Newton's method. In order to apply the scheme on a test bench, the maximum number of Newton iterations is set to 20 steps.

For ODEs, the convergence rates are for example derived in [7]. For higher index DAEs, results can be found in [17]. For DAEs, the convergence orders for differential variables and algebraic variables might be different. Since the solution of the servo-constraints inverse model is used as feedforward control, the algebraic variable $\mathbf{u}$ is of main interest here. In [17], the convergence rate for the algebraic variable for a s-step scheme with $s \geq 2$ is given as $s - 1$.

Tab. 1: Butcher tableau for the applied Runge Kutta schemes.

Impl. Euler ($s = 1$)	Radau IIA of order 5 ($s = 3$)			
	$\frac{4-\sqrt{6}}{10}$	$\frac{88-7\sqrt{6}}{360}$	$\frac{296-169\sqrt{6}}{1800}$	$\frac{-2+3\sqrt{6}}{225}$
	$\frac{4+\sqrt{6}}{10}$	$\frac{296-169\sqrt{6}}{1800}$	$\frac{88+7\sqrt{6}}{360}$	$\frac{-2-3\sqrt{6}}{225}$
1 | 1	1	$\frac{16-\sqrt{6}}{36}$	$\frac{16+\sqrt{6}}{36}$	$\frac{1}{9}$
| 1		$\frac{16-\sqrt{6}}{36}$	$\frac{16+\sqrt{6}}{36}$	$\frac{1}{9}$

BDF methods

Backwards differences formulas (BDF) are multi-step solvers, that use the last k solutions $\mathbf{x}_{n-1}$, $\mathbf{x}_{n-2}$, ..., $\mathbf{x}_{n-k}$ in order to calculate the next solution $\mathbf{x}_n$. Following [16], the BDF schemes for DAEs in implicit form of Eq. (12) are given by

$$\mathbf{F}\left(\mathbf{x}_n, \frac{1}{h}\sum_{i=0}^{k} \alpha_i\, \mathbf{x}_{n-i}, t_n\right) = \mathbf{0}\,, \tag{16}$$

with the coefficients α_i summarized in Tab. 2. In the beginning, the first k initial values must be provided e.g. using a single-step integration scheme. For DAEs with differentiation index 3, the following convergence result is stated in [16]. In case the first k values are consistent of order $k+1$ and the algebraic equations are solved at each step with accuracy $O\left(h^{k+3}\right)$ for $k = 1$ or accuracy $O\left(h^{k+2}\right)$ for $k \geq 2$ respectively, then the constant step size k-step BDF method converges with order k for $k < 7$.

Tab. 2: Parameter α_i for the k-step BDF scheme.

k	α_0	α_1	α_2	α_3	α_4	α_5	α_6
1	1	-1					
2	$\frac{3}{2}$	-2	$\frac{1}{2}$				
3	$\frac{11}{6}$	-3	$\frac{3}{2}$	$-\frac{1}{3}$			
4	$\frac{25}{12}$	-4	3	$-\frac{4}{3}$	$\frac{1}{4}$		
5	$\frac{137}{60}$	-5	5	$-\frac{10}{3}$	$\frac{5}{4}$	$-\frac{1}{5}$	
6	$\frac{49}{20}$	-6	$\frac{15}{2}$	$-\frac{20}{3}$	$\frac{15}{4}$	$-\frac{6}{5}$	$\frac{1}{6}$

Numerical Improvements

Accuracy and speed of the described integration methods might be further improved. The main bottle-neck of the schemes are the Newton iterations for solving the nonlinear Eq. (13) or Eq. (16) respectively. For higher index DAEs, the arising Jacobian matrix is ill-conditioned which can yield numerical rounding errors even for reasonable time steps. Also, the $m+1$ functions evaluations necessary to compute the Jacobian in each evaluation step for a nonlinear system of size m by a difference quotient make the method slow. Therefore, two methods are applied to improve numerical performance.

First, ill-conditioning of the Jacobian matrix is improved by scaling the algebraic equations with constants in the order of the time step size h as proposed in [16].

Moreover, to speed up Newton's method, the Jacobian is not recalculated in each step with difference quotients. Instead, Broyden's method is applied [18]. In that case, the Jacobian $\mathbf{J}$ of the nonlinear equations is approximated by $\tilde{\mathbf{J}}$. The approximation is updated in each Newton step within the iterative scheme

$$\boldsymbol{\eta}^i = -\tilde{\mathbf{J}}^{-1}\mathbf{F}_{\text{newton}}\left(\boldsymbol{\xi}^i\right) \tag{17}$$

$$\boldsymbol{\xi}^{i+1} = \boldsymbol{\xi}^i + \boldsymbol{\eta}^i \tag{18}$$

$$\tilde{\mathbf{J}}_{i+1} = \tilde{\mathbf{J}}_i + \frac{\mathbf{F}_{\text{newton}}\left(\boldsymbol{\xi}^{i+1}\right)\left(\boldsymbol{\eta}^i\right)^{\mathrm{T}}}{\|\boldsymbol{\eta}^i\|^2}, \tag{19}$$

where $\mathbf{F}_{\text{newton}}(\boldsymbol{\xi}) = \mathbf{0}$ denotes the nonlinear system of equations to be solved for $\boldsymbol{\xi} \in \mathbb{R}^m$ and i is the iteration step of Newton's method. The initial value $\tilde{\mathbf{J}}_0$ of the approximated Jacobian is obtained from the value of $\tilde{\mathbf{J}}$ in the last time step. The very first approximation is obtained by a difference quotient. In Section 3, applying Broyden's method continuously is compared to restarting the iterative scheme every 10 or 50 integration steps respectively, with a reevaluated Jacobian in order to stay close to the solution. Using Broyden's iterative scheme requires only 1 function evaluation instead of $m+1$ function evaluations in each Newton iteration.

3 Numerical and Experimental Results

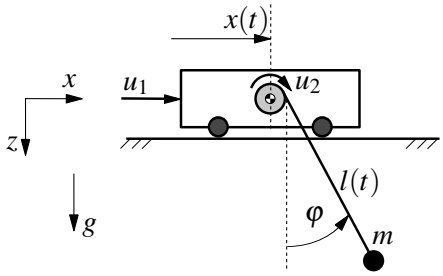
The solvers introduced in Sec. 2.3 are compared in numerical and experimental studies with respect to real-time performance and accuracy. First, convergence results are studied for an overhead crane. This is a typical example for a differentially flat underactuated multibody system. Secondly, the solvers are applied to a mass-on-car system in oder to analyze damping effects.

3.1 Overhead Crane

The overhead crane is a typical example in control systems. The overhead crane considered here consists of a trolley moving on parallel axes and a mass that is suspended by originally four ropes. The corresponding experimental setup is shown in Fig. 2(a). In the experiments shown here, all four ropes are actuated synchronously. Therefore,

the overhead crane is modeled with a single rope suspending the load. The model is shown in Fig. 2(b). It has $n = 3$ degrees of freedom and $n_\mathrm{u} = 2$ inputs which are collected in the vector $\mathbf{u} = \begin{bmatrix} u_1 & u_2 \end{bmatrix}^\mathrm{T}$. The first input u_1 actuates the trolley, while the second one u_2 actuates the winch on the trolley and therefore changes the rope length. Both inputs are considered as set point velocities.

(a) Experimental setup.



(b) Underactuated multibody system.

Fig. 2: Experimental setup and mechanical model of the overhead crane.

The trolley position is described by the coordinate x and the rope length is described by l. The unactuated swing angle is denoted by φ. The vector of minimal coordinates is chosen as

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_\mathrm{a} \\ \hdashline \mathbf{y}_\mathrm{u} \end{bmatrix} = \begin{bmatrix} x \\ l \\ \hdashline \varphi \end{bmatrix}, \qquad \mathbf{v} = \begin{bmatrix} \mathbf{v}_\mathrm{a} \\ \hdashline \mathbf{v}_\mathrm{u} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{l} \\ \hdashline \dot{\varphi} \end{bmatrix}, \tag{20}$$

where the dashed lines indicate the separation into actuated and unactuated variables. The equations of motion in ODE form arise as

$$\dot{\mathbf{y}} = \mathbf{v} \tag{21}$$

$$\left[\begin{array}{cc:c} \tau_1 & 0 & 0 \\ 0 & \tau_2 & 0 \\ \hdashline \cos\varphi & 0 & l \end{array}\right] \dot{\mathbf{v}} + \begin{bmatrix} \dot{x} \\ \dot{l} \\ \hdashline 2\,\dot{\varphi}\,\dot{l} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \hdashline -g\sin\varphi \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \hdashline 0 & 0 \end{bmatrix} \mathbf{u}, \tag{22}$$

where the actuators are modeled as first order systems with time constants τ_1 and τ_2 respectively. Considering redundant coordinates x_C and z_C for the load position in x- and z-direction, yields the redundant coordinates

$$\mathbf{y}_\mathrm{r} = \begin{bmatrix} \mathbf{y}_\mathrm{r,a} \\ \hdashline \mathbf{y}_\mathrm{r,u} \end{bmatrix} = \begin{bmatrix} x \\ l \\ \hdashline x_\mathrm{C} \\ z_\mathrm{C} \end{bmatrix}, \qquad \mathbf{v}_\mathrm{r} = \begin{bmatrix} \mathbf{v}_\mathrm{r,a} \\ \hdashline \mathbf{v}_\mathrm{r,u} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{l} \\ \hdashline \dot{x}_\mathrm{C} \\ \dot{z}_\mathrm{C} \end{bmatrix}. \tag{23}$$

The equations of motion in DAE form are then

$$\dot{\mathbf{y}}_\mathrm{r} = \mathbf{v}_\mathrm{r} \tag{24}$$

$$\left[\begin{array}{cc:cc} \tau_1 & 0 & 0 & 0 \\ 0 & \tau_2 & 0 & 0 \\ \hdashline 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right] \dot{\mathbf{v}}_\mathrm{r} + \begin{bmatrix} \dot{x} \\ \dot{l} \\ \hdashline 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \hdashline 0 \\ g \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \hdashline 2\,(x_\mathrm{C} - x) \\ 2 z_\mathrm{C} \end{bmatrix} \lambda + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \hdashline 0 & 0 \\ 0 & 0 \end{bmatrix} \mathbf{u} \tag{25}$$

$$z_\mathrm{C}^2 + (x_\mathrm{C} - x)^2 - l^2 = 0. \tag{26}$$

The system output is defined as the load position either represented in minimal or redundant coordinates

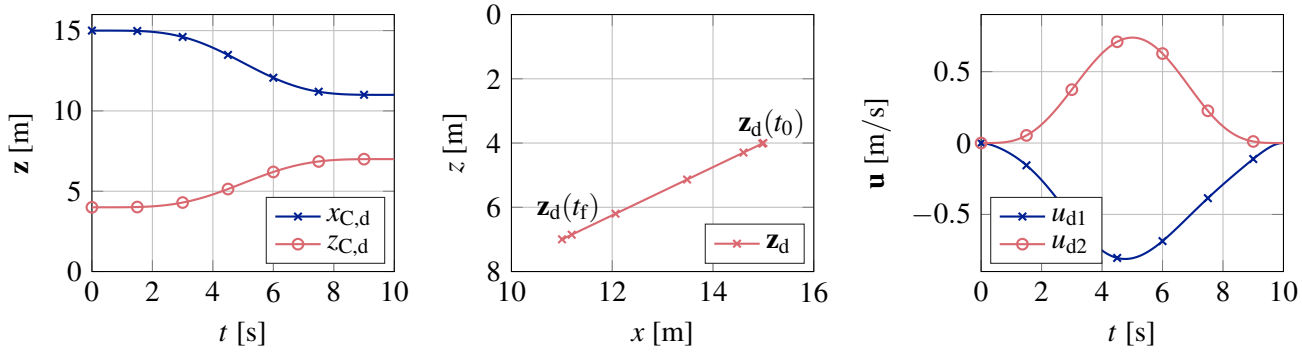
$$\mathbf{z} = \begin{bmatrix} x + l\sin\varphi \\ l\cos\varphi \end{bmatrix} = \begin{bmatrix} x_\mathrm{C} \\ z_\mathrm{C} \end{bmatrix}. \tag{27}$$

Accordingly, the servo-constraints are defined as

$$\mathbf{c} = \mathbf{z} - \mathbf{z}_\mathrm{d}(t) = \mathbf{z} - \begin{bmatrix} x_\mathrm{C,d}(t) \\ z_\mathrm{C,d}(t) \end{bmatrix}. \tag{28}$$

The inverse model problem represented by servo-constraints thus consists of either Eqs. (21)-(22) or Eqs. (24)-(26) together with Eq. (28). The resulting DAEs are of differentiation index 5. Here, the index is reduced to index 3 using the projection method proposed in [3] and by using the redundant coordinate formulation and directly substituting the servo-constraint Eq. (28) into the dynamics of Eqs. (24)-(26) as proposed in [4]. Both formulations are solved with the solvers reviewed in Section 2.3.

The desired output trajectory $\mathbf{z}_\mathrm{d}(t)$ is defined as a four times continuously differentiable polynomial and is shown in Fig. 3(a). Thereby, the desired trajectory is shown as a function of time and in the x,z plane in Fig. 3(a) and in Fig. 3(b) respectively. The desired control input $\mathbf{u}_\mathrm{d}$ as solution of the servo-constraints problem is shown in Fig. 3(c). The simulation parameters, which are identified on the test bench, are summarized in Tab. 3.



(a) Desired output $\mathbf{z}_\mathrm{d}(t)$. (b) Output $\mathbf{z}_\mathrm{d}(t)$ in the x,z plane. (c) Computed system input $\mathbf{u}_\mathrm{d}(t)$.

Fig. 3: Desired trajectory $\mathbf{z}_\mathrm{d}(t)$ and numerical results for the overhead crane with step size $h = 10\,\mathrm{ms}$.

Since the considered crane model is differentially flat, an analytical solution $\mathbf{u}_\mathrm{flat}(t)$ of the inverse model is available and for example derived in [9]. It is used as reference for the numerical solution obtained from solving the servo-constraints problem. The numerical solution $\mathbf{u}_\mathrm{d}(t)$ is compared to the analytical solution $\mathbf{u}_\mathrm{flat}(t)$ and the maximum error

$$e_\mathrm{max} = \max_t \left(\max_i |u_{\mathrm{d},i}(t) - u_{\mathrm{flat},i}(t)| \right) \tag{29}$$

is computed for comparison. Note that due to differential flatness, the inverse model does not include dynamics.

3.1.1 Analysis of Scaling the Algebraic Equations

First, scaling of the algebraic equations with a scalar factor ρ is analyzed. All following results are shown for the trajectory shown in Fig. 3. Results are discussed for the inverse model formulation using redundant coordinates

Tab. 3: Parameters of the overhead crane model.

Parameter	m	g	τ_1	τ_2
Value	18.9 kg	9.81 m/s^2	0.03 s	0.02 s

of Eqs. (24)-(26) and substituting the servo-constraint Eqs. (28) into the equations of motion as proposed in [4]. This yields an index 3 DAE formulation. The BDF scheme is here applied with $k = 4$ and is compared to Radau5. Thereby, the Jacobian matrix is approximated with Broyden's method as reviewed in Sec. 2.3 and the iterative scheme is reinitialized every 10 integration steps. The maximum error $e_{\max}$ as defined in Eq. (29) is shown for different step sizes h and different scaling factors $\rho \in \left[1, \frac{1}{h}, \frac{1}{h^2}, \frac{1}{h^3}\right]$ in Fig. 4. Thereby, the results for the BDF method are shown on the left and the results for Radau5 are shown in the right. A scaling of $\rho = 1$ corresponds to the original unscaled algebraic equations.

Both the results for the BDF and Radau5 scheme show that solving the unscaled DAEs with $\rho = 1$ yields an ill-conditioned system of equations. Both solutions show influences of rounding errors for step sizes $h < 0.1\,\mathrm{s}$ and $h < 0.05\,\mathrm{s}$ for BDF and Radau5, respectively.

For the BDF scheme, a scaling improves the numerical solution accuracy. Best results are obtained for the scaling $\rho = \frac{1}{h^3}$. However, note that the difference between scalings of $\rho = \frac{1}{h^2}$ and $\rho = \frac{1}{h^3}$ is very small. For these two scalings, the numerical solution begins to get unstable for step sizes $h < 10\,\mathrm{ms}$.

For Radau5, the best results are obtained for scaling the algebraic equations with $\rho = \frac{1}{h^2}$. The numerical solution begins to become inaccurate for step sizes $h < 5\,\mathrm{ms}$. Note that in contrast to the BDF results, a scaling of $\rho = \frac{1}{h^3}$ strongly reduces the solution accuracy. In the following, both methods will be applied with a scaling of $\rho = \frac{1}{h^2}$.

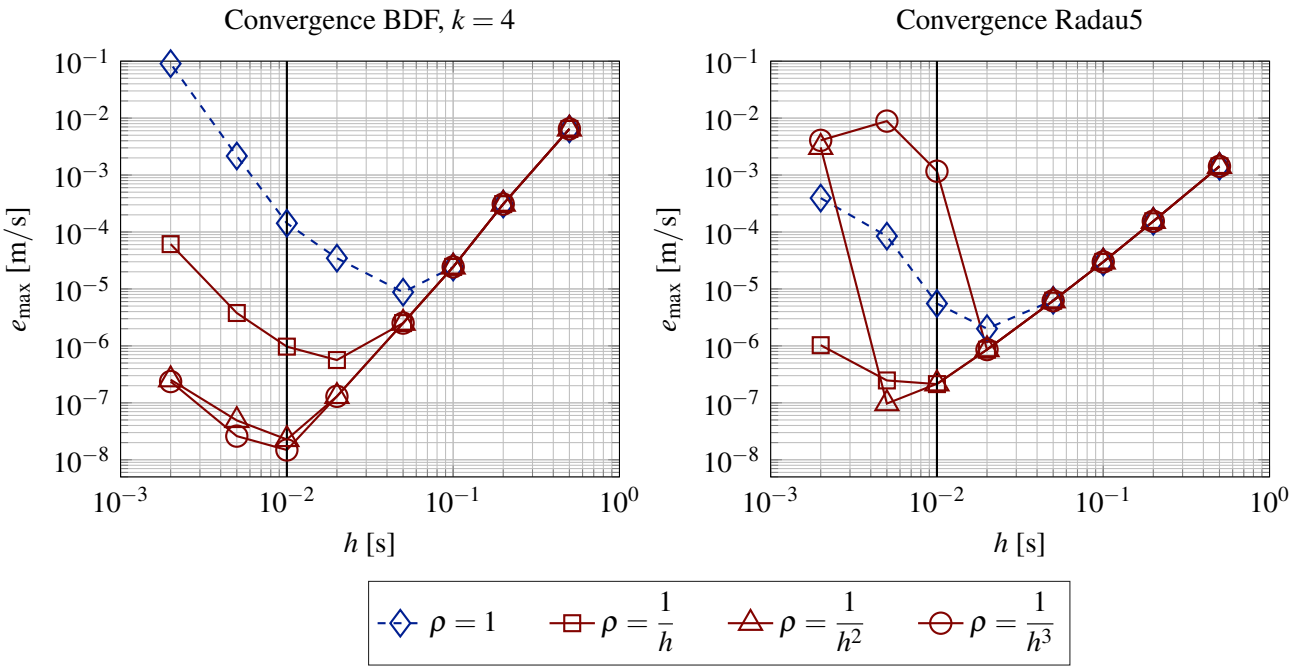


Fig. 4: Variation of scaling ρ of the algebraic equations.

3.1.2 Analysis of Radau5 Scheme with Broyden Update

Next, the application of Broyden's method, as reviewed in Section 2.3, is analyzed. Results are discussed again for the inverse model formulation using redundant coordinates of Eqs. (24)-(26) and substituting the servo-constraint Eqs. (28) into the equations of motion as proposed in [4]. Broyden's method is applied in the Radau5 scheme for solving the arising set of nonlinear Eqs. (13). Thereby, Broyden's method is initialized with calculating the Jacobian $\mathbf{J}$ with a difference quotient. The results of applying Broyden's method continuously are denoted by $\tilde{\mathbf{J}}$. They are compared to reinitializing Broyden's iteration every 10 and every 50 steps, denoted by $\tilde{\mathbf{J}}_{10}$ and $\tilde{\mathbf{J}}_{50}$ respectively. As reference, the results are also given for regular Newton iterations with recalculating the Jacobian in every iteration step, denoted by $\mathbf{J}$.

The maximum error $e_{\max}$ as defined in Eq. (29) is shown for different step sizes h on the left in Fig. 5. Moreover, the total calculation time t_{tot} on a standard desktop computer for the complete simulation of the 10 s trajectory described above is shown over the step size h in the right hand plot of Fig. 5. Note that the total calculation time only gives an indication of the real-time capability, since it shows if the average computation time $t_{\text{calc},i}$ of one integration step i is faster than the step size h. A single integration step might still be slower than the time step size. In the discussion of the experiments in Sec. 3.1.5, the computation times $t_{\text{calc},i}$ for each time step will be compared. The 10 s simulation time and the step size $h = 10\,\text{ms}$ available on the test bench are denoted by thick lines in the plots.

The convergence diagram shows that the Radau5 scheme converges with an order of approximately 2, which corresponds to the convergence results stated in Section 2.3. This convergence is independent of the computation method of the Jacobian for step sizes $h \geq 10\,\text{ms}$. For smaller step sizes, the results show that recalculating the Jacobian $\mathbf{J}$ in every step yields the most accurate solution. This is reasonable because the Jacobian $\mathbf{J}$ is the most accurate one. Numerical rounding errors start to influence the solution for time steps in the size of $h = 2\,\text{ms}$. Since recalculating the Jacobian requires many function evaluations, the calculation is not real-time capable for time steps smaller than $h < 20\,\text{ms}$ as can be read off the right hand diagram.

Broyden's method, denoted by $\tilde{\mathbf{J}}$ does not require as many function evaluations and is therefore much faster for the same error $e_{\max}$. However, the convergence diagram shows that the solution is ill-conditioned for step sizes $h \leq 10\,\text{ms}$. The approximated Jacobian $\tilde{\mathbf{J}}$ becomes inaccurate after too many iterations. Reinitializing Broyden's method after 10 or 50 iteration steps yields accurate results up to step sizes of $h = 5\,\text{ms}$ respectively. At the step size $h = 10\,\text{ms}$, the solution calculated with Broyden's Jacobian $\tilde{\mathbf{J}}_{10}$ is approximately 9 times faster compared to recalculating the Jacobian $\mathbf{J}$ in every step. The calculation times are $t_{\text{tot}}\left(\tilde{\mathbf{J}}_{10}\right) \approx 2.4\,\text{s}$ and $t_{\text{tot}}(\mathbf{J}) \approx 20.5\,\text{s}$. Results for convergence might be improved by allowing more Newton iterations. However, this will slow down the simulation and might not be possible in real-time. Therefore, the maximum number of Newton iterations is set to 20 iterations per integration step.

Due to different hardware on the test bench, the exact calculation times are different on the laboratory setup. However, the qualitative behavior of the different schemes stays the same. The experiments will be performed using Broyden's method and reinitializing it every 10 integration steps.

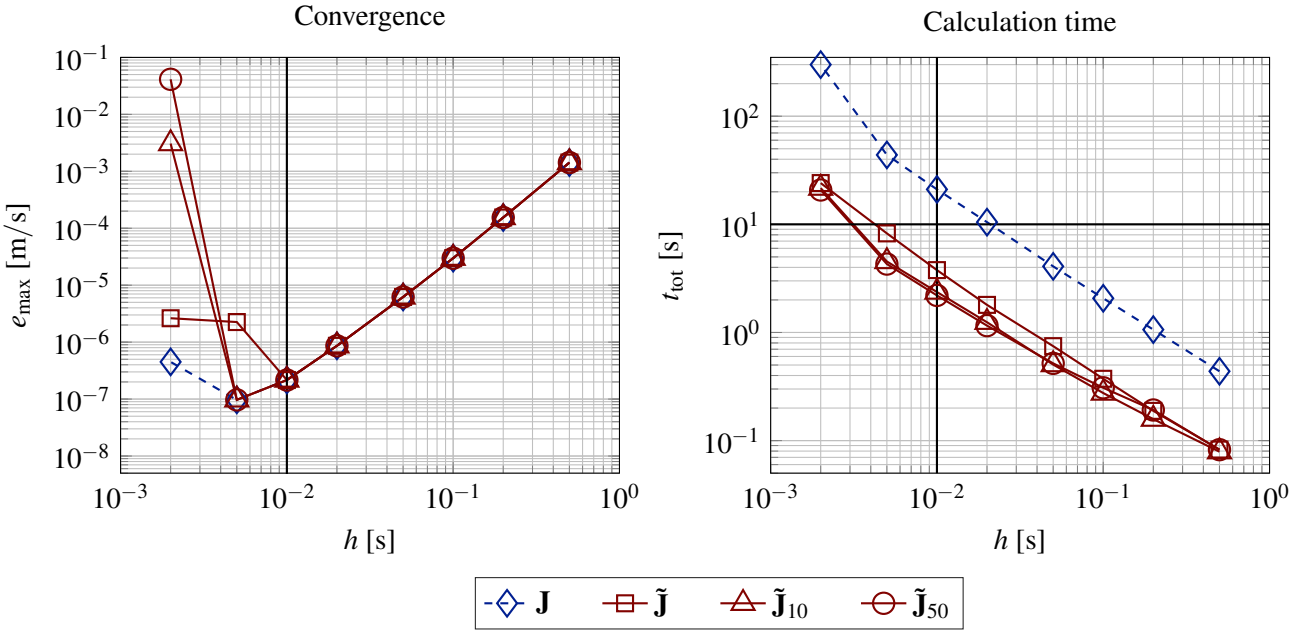


Fig. 5: Variation of Broyden's method within the Radau5 scheme.

3.1.3 Analysis of BDF Scheme

Convergence of the BDF schemes, as described in Section 2.3, is analyzed by varying the number of steps k and calculating the maximum error $e_{\max}$ for different time steps. The simulations are performed for the same trajectory and the same index 3 inverse model formulation as before. Note that for the nonlinear Eqs. (16) arising in the BDF scheme, Newton's method is used with recalculating the Jacobian $\mathbf{J}$ in every time step. Results are shown in Fig. 6 and are compared to the previously analyzed Radau5 scheme using $\tilde{\mathbf{J}}_{10}$.

The convergence plot validates the convergence orders stated in Section 2.3 for $k \in [1,2,...,6]$. Note that the maximum error $e_{\max}$ for $k > 2$ is orders of magnitude smaller compared to the implicit Euler obtained from $k = 1$, which is usually used in the context of servo-constraints. For step sizes $k \geq 5$, the problem is ill-conditioned and numerical rounding errors start to influence the solution for step sizes $h < 20\,\text{ms}$ and $h < 50\,\text{ms}$ for $k = 5$ and $k = 6$ respectively. A selection of $k = 4$ yields convergence of order 4 and thus higher order convergence compared to the Radau5 scheme, denoted by the blue dashed lines in the convergence diagram. Moreover, the solution with $k = 4$ yields numerical stable results for the test bench step size. Looking at the calculation time t_{tot} in the right hand diagram, the Radau5 scheme and the BDF schemes yield comparable calculation times. The BDF scheme is real-time capable for step sizes $h > 2\,\text{ms}$ for all k. Due to stability in the order of the test bench step size and real-time applicability, the BDF scheme with $k = 4$ is chosen for the experiments.

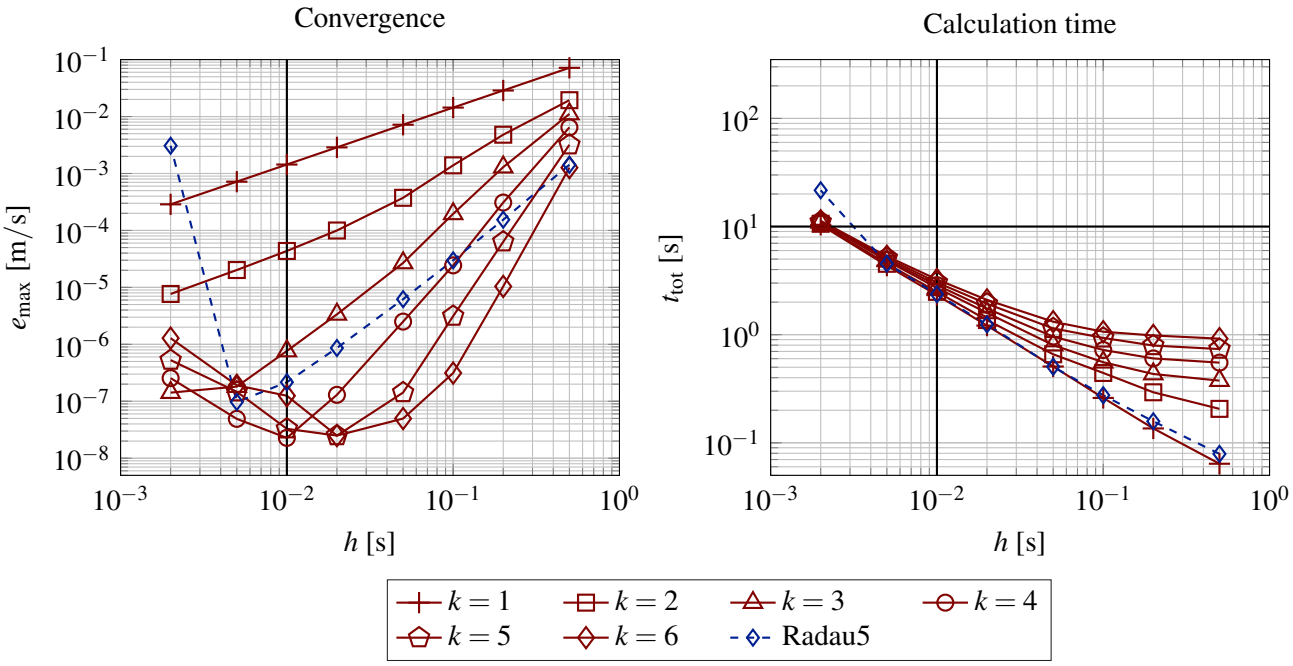


Fig. 6: Variation of steps k in the BDF scheme compared to Radau5.

3.1.4 Analysis of Index Reduction Methods

So far, the index 3 formulation was used that is obtained from directly substituting the servo-constraints into the dynamic equations. This formulation is now denoted by *subs* and is compared to the index 3 formulation obtained from the projection approach proposed in [3]. Thereby, the dynamic Eqs. (21)-(22) in minimal coordinates are projected onto the constrained and unconstrained manifold. This formulation is denoted by *proj*. Both formulations are solved using the BDF scheme with $k = 4$ and using Radau5 with the Jacobian approximation $\tilde{\mathbf{J}}_{10}$. The convergence diagram and calculation time is shown in Fig. 7.

All formulations converge for step sizes $h \geq 10\,\text{ms}$, while the *subs* formulations solved with Radau5 and BDF show slightly higher convergence rates compared to the *proj* formulation respectively. Larger differences are visible in the calculation time diagram. The total calculation time of the *subs* formulation is 1.5 and 2 times faster

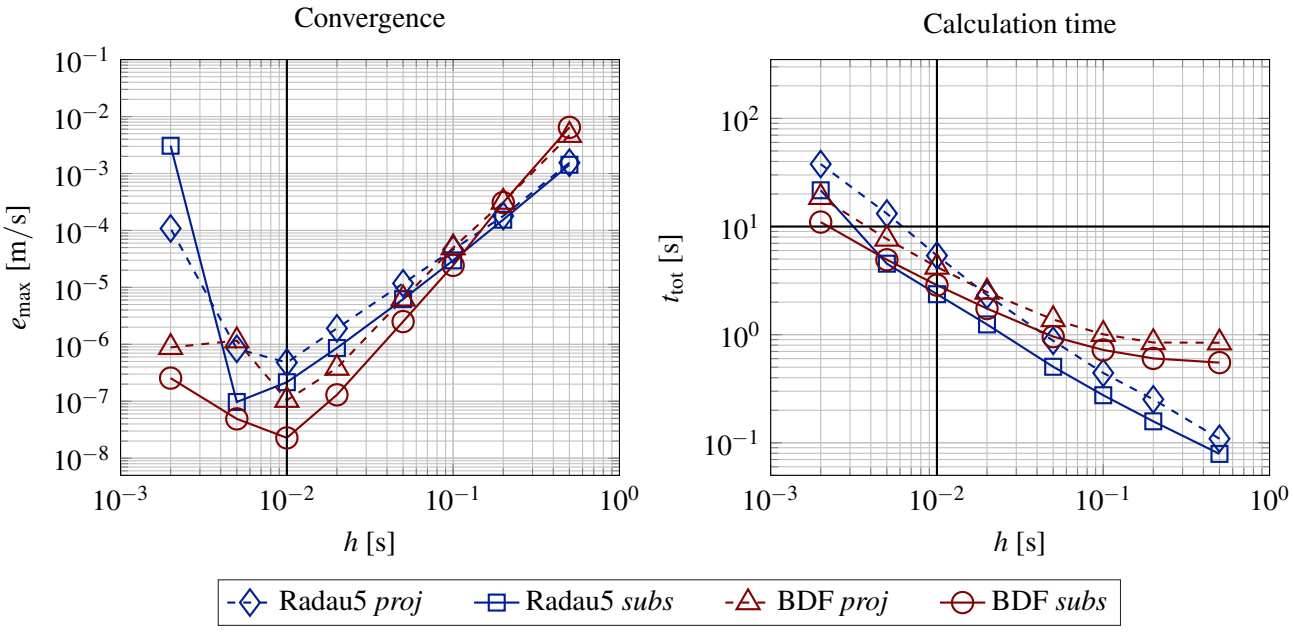


Fig. 7: Variation of the servo-constraints problem formulation.

compared to the *proj* formulation respectively for the BDF and Radau5 scheme. This is due to a smaller system of equations for the *subs* formulation. At test bench step size $h = 10\,\text{ms}$, the *subs* solution with total calculation times of $t_{\text{tot}} \approx 2.9\,\text{s}$ and $t_{\text{tot}} \approx 2.4\,\text{s}$ for BDF and Radau5 respectively is well in the real-time capable regime. From the results is concluded that the *subs* formulation is slightly better conditioned and it is therefore used in the experiments. Note that the absolute errors in the order of $e_{\max} \approx 1 \times 10^{-7}\,\text{m/s}$ are very small compared to other expected disturbances in the actuators such as friction.

3.1.5 Experimental Results

Experiments are carried out on the laboratory setup shown in Fig. 2(a). Its workspace dimensions are $x \in [10, ..., 23]\,\text{m}$ and $l \in [3, ..., 9]\,\text{m}$. The solvers are exported to a LABVIEW program which is controlling the test bench. Therefore, Matlab code is exported using the c-code generator. The solver is called in form of a dynamic link library (DLL) in every control loop iteration running with step size $h = 10\,\text{ms}$. The time the LABVIEW program spends inside the DLL in every iteration is measured by a microsecond clock and is denoted by $t_{\text{calc},i}$ for every control loop iteration i.

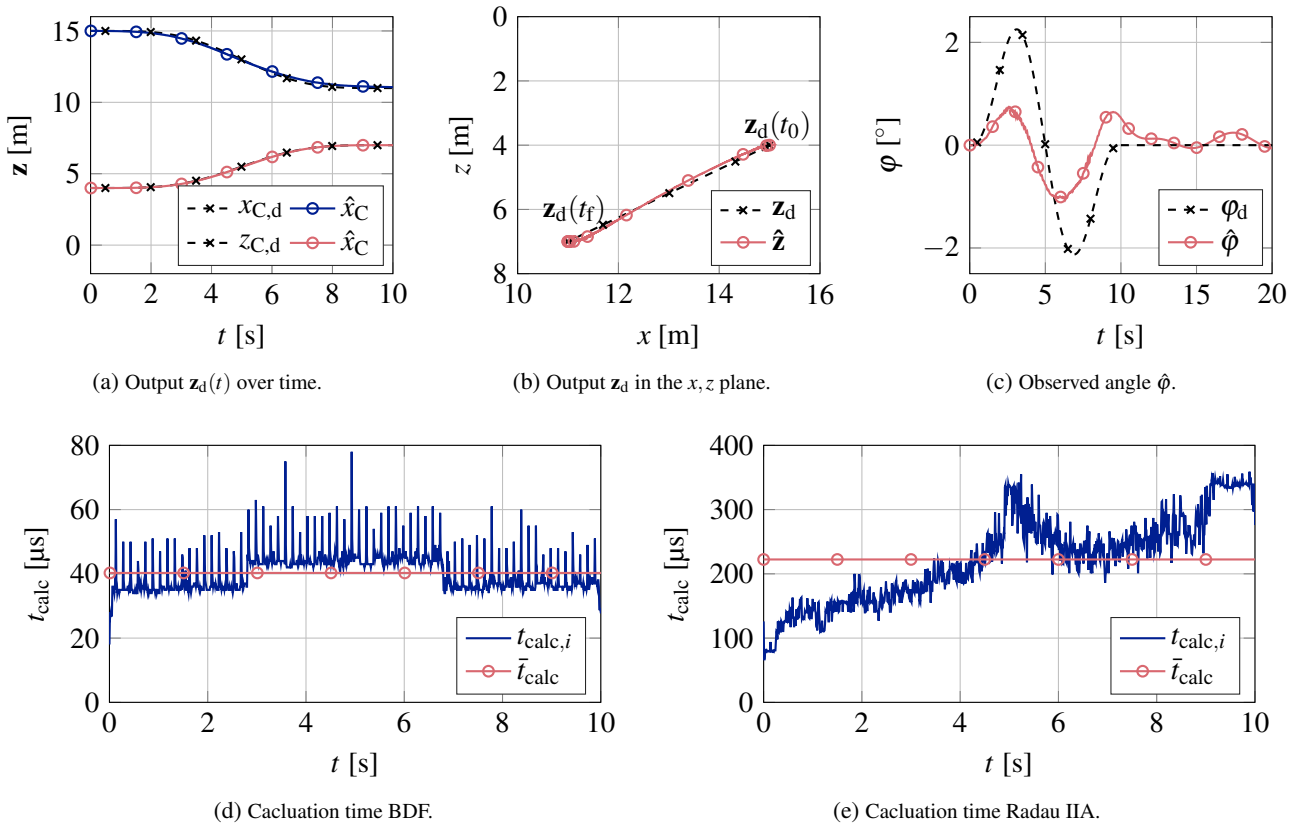
The experimental results for only using the inverse model based on servo-constraints are shown in Fig. 8 for the same trajectory as before. Thereby, the experimental data is denoted by $\hat{\square}$. Note that the swing angle $\hat{\varphi}$ of the load cannot be measured directly and is therefore observed by an unscented Kalman filter based on rope force measurements. The results in Figs 8(a) and 8(a) show that the feedforward controller is able to move the overhead crane on the desired trajectory very well. There are some minor tracking errors. Moreover, the load is not completely in rest at the end of the trajectory. A small swing angle of approximately $\hat{\varphi} \approx 0.3^\circ$ remains, see Fig. 8(c). Since there is no direct measurement of the swing angle, the tracking error and observer error cannot be separated. However, considering the size of the experimental setup and length of the trajectory, these errors are rated to be small. Any errors can be further minimized by adding a state feedback controller.

The calculation time $t_{\text{calc},i}$ of each integration step i is shown in Figs. 8(d) and 8(e) for the BDF as well as the Radau5 integrator respectively. Due to the previously discussed results, the BDF integrator is implemented with $k = 4$ and the Radau5 scheme is used with Broyden's method and the Jacobian $\tilde{\mathbf{J}}_{10}$. For the displayed experiment, the time $t_{\text{calc},i}$ spent in each iteration for solving the index 3 DAE with the BDF scheme varies in a region around a minimum of $t_{\text{calc},min} = 17\,\mu\text{s}$ and a maximum of $t_{\text{calc},max} = 78\,\mu\text{s}$, while the larger calculation times do not appear as often. This is reflected in the mean calculation time of $\bar{t}_{\text{calc}} = 40.28\,\mu\text{s}$. For the Radau5 solver, the calculation

time varies within $t_{\text{calc},min} = 67\,\mu\text{s}$ and $t_{\text{calc},max} = 359\,\mu\text{s}$ and the mean calculation time is $\bar{t}_{\text{calc}} = 222.38\,\mu\text{s}$.

The results are verified in a series of 5 experiments for each solver. For the BDF scheme, the overall mean calculation time is $\bar{t}_{\text{calc}} = 40.29\,\mu\text{s}$ with a standard deviation of $\sigma_{\text{calc}} = 0.08\,\mu\text{s}$. For the Radau5 scheme, the overall mean calculation time is $\bar{t}_{\text{calc}} = 235.87\,\mu\text{s}$ with a standard deviation of $\sigma_{\text{calc}} = 10.85\,\mu\text{s}$. The experimental results therefore confirm the numerical studies shown in Figs. 3-7.

Compared to the simulation results shown in Fig. 7, the computation times of the BDF and Radau5 scheme are much faster on the experimental setup due to c-code export and execution on a real-time operating system. In contrast to the simulations, the BDF scheme is approximately 6 times faster compared to Radau5. Note that the absolute calculation time strongly depends on the chosen implementation of the integration method and the internal optimization during c-code export. However, the main result is here that the integrators are fast enough to solve the index 3 DAEs in real-time. Considering the control loop step size $h = 10\,\text{ms}$, there is potential to solve even larger systems with more degrees of freedom, e.g. a three dimensional model.



(a) Output $\mathbf{z}_\text{d}(t)$ over time.

(b) Output $\mathbf{z}_\text{d}$ in the x,z plane.

(c) Observed angle $\hat{\varphi}$.

(d) Cacluation time BDF.

(e) Cacluation time Radau IIA.

Fig. 8: Experimental results for the overhead laboratory crane.

3.2 Mass-on-car System

The mass-on-car system is shown in Fig. 9 and consists of a car with mass m_1 moving horizontally and a mass m_2 that moves on a plane inclined by a fixed angle α. Both bodies are connected by a linear spring-damper combination with coefficients k and d, respectively. The system has 2 degrees of freedom and a single input, which is the force F. The system is analyzed in detail in [6] to show flat and non-flat configurations. Here, the focus is on the non-flat configuration with the angle $0^\circ < \alpha < 90^\circ$. The parameters are given in Tab. 4.

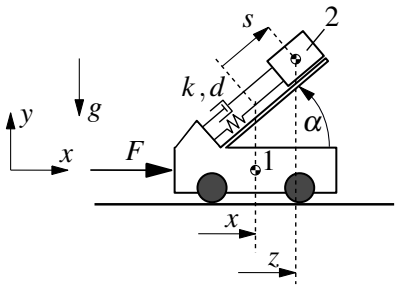


Fig. 9: Model of mass-on-car system.

Tab. 4: Parameters for mass-on-car system.

Parameter	m_1	m_2	k	d
Value	1 kg	1 kg	5 N/m	0 Ns/m , 1 Ns/m

The position of the car is denoted by x and the relative position of mass m_2 with respect to the car is denoted by s. Thus, the minimal coordinates are chosen as

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_\mathrm{a} \\ \hdashline \mathbf{y}_\mathrm{u} \end{bmatrix} = \begin{bmatrix} x \\ \hdashline s \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} \mathbf{v}_\mathrm{a} \\ \hdashline \mathbf{v}_\mathrm{u} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \hdashline \dot{s} \end{bmatrix}. \tag{30}$$

The equations of motion are

$$\dot{\mathbf{y}} = \mathbf{v} \tag{31}$$

$$\left[\begin{array}{c:c} m_1+m_2 & m_2\cos\alpha \\ \hdashline m_2\cos\alpha & m_2 \end{array}\right] \dot{\mathbf{v}} = -\begin{bmatrix} 0 \\ \hdashline k\,s + d\,\dot{s} \end{bmatrix} + \begin{bmatrix} 1 \\ \hdashline 0 \end{bmatrix} F, \tag{32}$$

with the scalar system input $u = F$ applied horizontally on the car. The scalar system output is the horizontal position of mass 2 with

$$z(\mathbf{y}) = x + s\cos\alpha\,. \tag{33}$$

For derivation of the inverse model, the servo-constraints are consequently chosen as

$$c(\mathbf{y},t) = x + s\cos\alpha - z_\mathrm{d}(t) \tag{34}$$

with the desired trajectory $z_\mathrm{d}(t)$. In the configuration with the angle $0^\circ < \alpha < 90^\circ$, the system is of relative degree $r = 2$. The respective inverse model DAE given by the combination of Eqs. (31)-(32) with Eq. (34) has therefore differentiation index 3 [6]. Due to relative degree $r = 2$ and a total of 4 states in state space, internal dynamics remains of dimension 2. This non-flat configuration is chosen because the internal dynamics is stable and of second order. Thus, by choosing suitable parameters which yield complex eigenvalues, the internal dynamics is able to oscillate without damping. Damping properties of the solvers can be easily compared for this configuration. As stated in Sec. 2.2, the internal dynamics can be extracted in ODE form by a coordinate transformation. Choosing new coordinates

$$\tilde{\mathbf{y}} = \begin{bmatrix} z \\ s \end{bmatrix} \quad \text{and} \quad \tilde{\mathbf{x}} = \begin{bmatrix} \tilde{\mathbf{y}} \\ \dot{\tilde{\mathbf{y}}} \end{bmatrix}, \tag{35}$$

and transforming the Eqs. (31)-(32) to the new coordinates $\tilde{\mathbf{x}}$ yields the first order dynamics

$$\dot{\tilde{\mathbf{x}}} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -\dfrac{m_1\cos\alpha}{a\,m_2}\,k & 0 & -\dfrac{m_1\cos\alpha}{a\,m_2}\,d \\ 0 & -\dfrac{m_1+m_2}{a\,m_2}\,k & 0 & -\dfrac{m_1+m_2}{a\,m_2}\,d \end{bmatrix} \tilde{\mathbf{x}} + \begin{bmatrix} 0 \\ 0 \\ \dfrac{\sin^2\alpha}{a} \\ \dfrac{-\cos\alpha}{a} \end{bmatrix} F \tag{36}$$

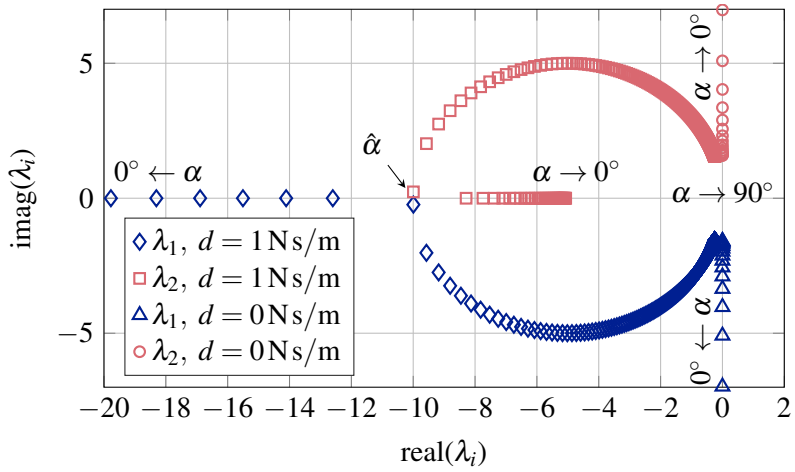


Fig. 10: Eigenvalues λ_i of the internal dynamics in the coordinate s of the mass-on-car system.

with the coefficient $a = m_1 + m_2 \sin^2\alpha$. The third line of Eq. (36) can be solved for the desired control input F with a desired trajectory $z_\mathrm{d}(t)$, $\ddot{z}_\mathrm{d}(t)$ such that

$$F_\mathrm{d}(t) = \frac{a}{\sin^2\alpha}\,\ddot{z}_\mathrm{d}(t) + \frac{m_1\cos\alpha}{m_2\,\sin^2\alpha}\,(ks + d\dot{s})\,. \tag{37}$$

Using this information in the fourth line of Eq. (36) extracts the internal dynamics in ODE form as

$$\left(m_2\sin^2\alpha\right)\ddot{s} + d\dot{s} + ks = -\ddot{z}_\mathrm{d}(t)\,m_2\cos\alpha\,. \tag{38}$$

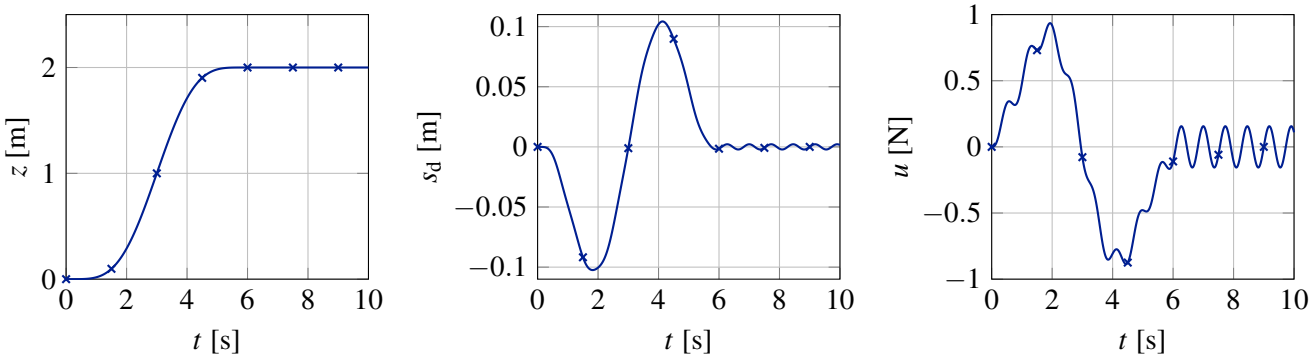
This internal dynamics is driven by the desired output $\ddot{z}_\mathrm{d}(t)$. The zero dynamics is given by setting the output identically to zero $z_\mathrm{d}(t) = \dot{z}_\mathrm{d}(t) = \ddot{z}_\mathrm{d}(t) = 0\;\forall\,t$. Stability of the zero dynamics is shown in the root locus plot in Fig. 10. The system parameters are given in Tab. 4. For the undamped system with $d = 0\,\mathrm{N\,s/m}$, the eigenvalues λ_i of the internal dynamics of Eq. (38) stay on the imaginary axis. For the damped system, the eigenvalues are complex conjugates for angle $\alpha > \hat{\alpha}$. The breakaway point $\hat{\alpha}$ describing critical damping is at

$$\hat{\alpha} = \arcsin\left(\frac{d}{2\sqrt{k\,m_2}}\right). \tag{39}$$

For the chosen parameters holds $\hat{\alpha} \approx 12.9°$ for $d = 1\,\mathrm{N\,s/m}$. For smaller values $\alpha < \hat{\alpha}$, the eigenvalues λ_i become real and the system is strongly damped. For the following damping analysis, an angle of $\alpha = 15°$ and no damping $d = 0\,\mathrm{N\,s/m}$ is chosen, so that the internal dynamics is able to oscillate.

For the following simulations, a four times continuously differentiable polynomial which smoothly merges into a constant final position is chosen as desired trajectory $z_\mathrm{d}(t)$. It is visualized in Fig. 11(a). As reference, the ODE (38) of the internal dynamics is solved with the step size controlled Runge Kutta scheme of Dormand and Prince, implemented in Matlab as *ode45*. The solution is denoted by s_d. For the chosen desired trajectory, the internal dynamics is excited and oscillates after transition to the final position, see Fig.11(b). This oscillation is not observable from the output z as it is internal dynamics. The respective desired control input $u_\mathrm{d} = F_\mathrm{d}$ is obtained from substituting the internal dynamics solution s_d, $\dot{s}_\mathrm{d}$ into Eq. (37) and is shown in Fig. 11(c). In contrast to the previously shown flat overhead crane, the control input u_d does not rest at $u = 0\,\mathrm{N}$ at the end of the transition at time $t = 6\,\mathrm{s}$. The desired input oscillates around $u = 0\,\mathrm{N}$ to compensate for the oscillation of the internal dynamics.

In the following, the inverse model consisting of Eqs. (31)-(32) and Eq. (34) is solved in the original index 3 formulation with the different solvers. Applying the implicit Euler with step size $h = 10\,\mathrm{ms}$ illustrates significant damping of the internal dynamics oscillations. This is reflected by comparison of the calculated desired input over time in Fig. 12. Due to internal dynamics, the implicit Euler with step size $h = 10\,\mathrm{ms}$ cannot reflect the system behavior accurately. Moreover, applying the calculated system input in a forward time simulation shows



(a) Desired output $z_\mathrm{d}(t)$. (b) Internal Dynamics s_d. (c) Desired Input u_d.

Fig. 11: Desired trajectory $z_\mathrm{d}(t)$, respective internal dynamics s_d and desired input u_d for the mass-on-car system.

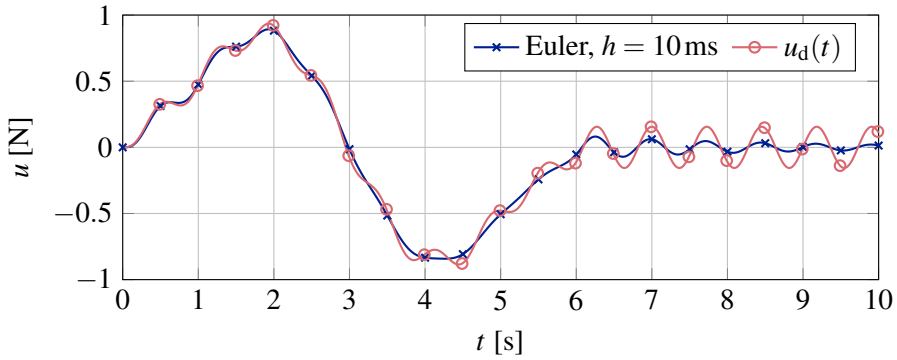


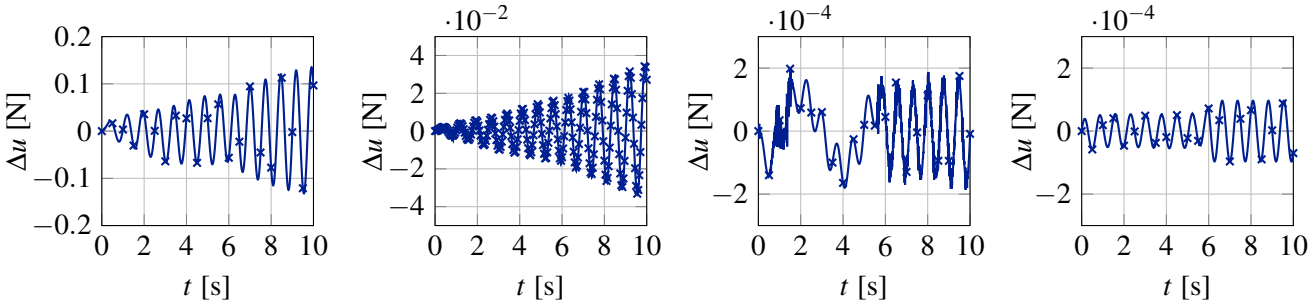
Fig. 12: Desired input $u_\mathrm{d}(t)$ and DAE solution for the implicit Euler with $h = 10\,\mathrm{ms}$.

oscillations in the system output z, which are not desirable. These tracking errors can be avoided by choosing a more sophisticated solver or a smaller step size.

Applying the implicit Euler with smaller step size of $h = 1\,\mathrm{ms}$ can reduce the error constant, but the dynamics is still damped. This is reflected in Fig. 13 by plotting the difference Δu between the reference solution obtained by solving Eq. (38) and the solution obtained from solving the inverse model DAE. For the implicit Euler with step sizes $h = 10\,\mathrm{ms}$ and $h = 1\,\mathrm{ms}$, the error Δu grows linearly in time. This is visualized in Figs. 13(a) and 13(b). In contrast, the energy conserving BDF and Radau5 solvers do not show a growing error Δu, which is shown in Figs. 13(c) and 13(d) respectively. Note that the BDF scheme is again applied with $k = 4$. For Radau5, the Jacobian is approximated with Broyden's method in configuration $\tilde{\mathbf{J}}_{10}$. The error of both schemes is two orders of magnitude smaller compared to the error of the implicit Euler. Therefore, these solvers should be preferred over the implicit Euler for solving the inverse model problem for multibody systems with internal dynamics.

4 Conclusion

For trajectory tracking problems of underactuated multibody systems, the servo-constraints approach can be implemented in a straightforward way. By formulating the desired output trajectory as constraint on the system dynamics, a set of higher index DAEs arises. These DAEs are solved for the desired system input and desired state trajectory. The desired input is then used as feedforward control. In order to apply the method on a real technical setup, the solution must be obtained online. In the context of servo-constraints, the DAEs are usually solved using the implicit Euler scheme. In this contribution, higher order schemes, such as a 3-step Runge Kutta scheme and BDF schemes are applied to solve the inverse model DAEs for different system classes. The results are compared to the implicit Euler.



(a) Error Δu for Euler, $h = 10\,\text{ms}$. (b) Error Δu for Euler, $h = 1\,\text{ms}$. (c) Error Δu for BDF, $h = 10\,\text{ms}$. (d) Error Δu for Radau5, $h = 10\,\text{ms}$.

Fig. 13: Input errors Δu compared to reference ODE solution given by Eq. (38).

For differentially flat systems, the implicit Euler can solve the inverse model quite well, because the inverse model does not have any dynamics. Using higher order schemes reduces the numerical error due to higher order convergence. Concerning application, the numerical error should be at least as small as other disturbances in the actuators, e.g. friction. The reviewed higher order schemes require more function evaluations and more computational effort compared to the implicit Euler scheme. The effort can be reduced by for example using Broyden's method to approximate the Jacobian of the set of nonlinear equations instead of recalculating it in every integration step. Moreover, scaling of the algebraic equation improves conditioning of the Jacobian and therefore the numerical solution. Experimental data on a laboratory overhead crane shows that also the higher order schemes can solve the DAEs online.

For systems with non-flat output and stable internal dynamics, the inverse model itself is a dynamical system. The internal dynamics must be analyzed with respect to stability in order to apply the servo-constraints approach. It is shown that the implicit Euler might damp the internal dynamics substantially. Therefore, the solver cannot represent the model behavior accurately and does not provide a good inverse model. Higher order schemes do not show such a numerical damping and should be preferred over the implicit Euler for such systems.

References

[1] A. Isidori, *Nonlinear control systems*. Berlin [u.a.]: Springer, 3. ed., 1996.

[2] T. Berger, "The zero dynamics form for nonlinear differential-algebraic systems," *Automatic Control, IEEE Transactions on*, vol. 62, no. 8, pp. 4131–4137, 2017.

[3] W. Blajer and K. Kolodziejczyk, "A geometric approach to solving problems of control constraints: Theory and a DAE framework," *Multibody System Dynamics*, vol. 11, no. 4, pp. 343–364, 2004.

[4] S. Otto and R. Seifried, "Real-time trajectory control of an overhead crane using servo-constraints," *Multibody System Dynamics*, vol. 42, pp. 1–17, Jan 2018.

[5] S. L. Campbell, "High-index differential algebraic equations," *Mechanics of Structures and Machines*, vol. 23, no. 2, pp. 199–222, 1995.

[6] R. Seifried and W. Blajer, "Analysis of servo-constraint problems for underactuated multibody systems," *Mechanical Sciences*, vol. 4, no. 1, pp. 113–129, 2013.

[7] E. Hairer, *Stiff and differential-algebraic problems*. Hairer, Ernst Solving ordinary differential equations / E. Hairer; G. Wanner, Berlin u.a.: Springer, 2., rev. ed., corrected 2. printing ed., 2002.

[8] E. Hairer, *Nonstiff problems*. Hairer, Ernst Solving ordinary differential equations / E. Hairer, Berlin u.a.: Springer, 2., rev. ed., 3. printing ed., 2008.

[9] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "Flatness and defect of non-linear systems: introductory theory and examples," *International Journal of Control*, vol. 61, no. 6, pp. 1327–1361, 1995.

[10] G. Bastos, R. Seifried, and O. Brüls, "Analysis of stable model inversion methods for constrained underactuated mechanical systems," *Mechanism and Machine Theory*, vol. 111, no. Supplement C, pp. 99 – 117, 2017.

[11] R. Altmann, P. Betsch, and Y. Yang, "Index reduction by minimal extension for the inverse dynamics simulation of cranes," *Multibody System Dynamics*, vol. 36, no. 3, pp. 295–321, 2016.

[12] R. Altmann and J. Heiland, "Simulation of multibody systems with servo constraints through optimal control," *Multibody System Dynamics*, pp. 1–24, 2016.

[13] S. L. Campbell and C. W. Gear, "The index of general nonlinear daes," *Numerische Mathematik*, vol. 72, pp. 173–196, Dec 1995.

[14] J. Rudolph, *Flatness Based Control of Distributed Parameter Systems*. Shaker, 2003.

[15] J.-J. E. Slotine and W. Li, *Applied nonlinear control*. Englewood Cliffs, NJ: Prentice Hall, 1991.

[16] K. E. Brenan, *Numerical solution of initial-value problems in differential-algebraic equations*. New York: North-Holland, 1989.

[17] E. Hairer, M. Roche, and C. Lubich, "The numerical solution of differential-algebraic systems by runge-kutta methods," 1989.

[18] M. Hermann, *Numerische Mathematik*. Berlin, Boston: de Gruyter, 2011.