MBSlib - An Efficient Multibody Systems Library for Kinematics and Dynamics Simulation, Optimization and Sensitivity Analysis

Janis Wojtusch, *Member, IEEE*, Jürgen Kunz, *Member, IEEE*, Oskar von Stryk, *Member, IEEE*

***Abstract*—The dynamic behavior of many technical and biomechanical systems can be modeled, simulated and optimized by using a multibody systems approach. Systems with many degrees of freedom typically result in complex and high-dimensional multibody systems models that necessitate capable modeling approaches and efficient computational algorithms as well as impede the numerical solution of related optimization problems. In this paper, the efficient and modular multibody systems library MBSLIB for kinematics and dynamics simulation, optimization and sensitivity analysis is presented. MBSLIB provides an intuitive modeling interface, modular software architecture, efficient computational algorithms as well as the computation of derivatives with respect to system states, control variables or model parameters. In combination with gradient-based optimization methods, the derivatives can be used to facilitate the numerical solution of optimization problems significantly. The library is open and can be obtained from the MBSLIB website. These features make MBSLIB a powerful, flexible and lightweight modeling, simulation and optimization library suitable for many technical and biomechanical applications.**

***Index Terms*—Direct/Inverse Dynamics Formulation, Optimization and Optimal Control, Calibration and Identification, Sensitivity Analysis, Multibody Systems.**

I. INTRODUCTION

DETAILED knowledge of the dynamic behavior is an important foundation for the study, design, improvement and control of technical and biomechanical systems [1], [2]. Many systems in robotics, assistance or sports applications including robot arms, e.g. [3], [4], humanoid robots, e.g., [5], [6], or the human locomotor system, e.g., [7], [8], can be modeled as multibody systems consisting of rigid links, multidimensional joints and mechanical components like springs, dampers, drives or muscle-tendon units. The dynamic behavior of the modeled system may then be studied by running kinematics and dynamics simulations to compute joint trajectories or joint forces and torques, applying model based or optimal control to find control and state trajectories for given performance criteria or performing sensitivity analysis to rate the influence of specific model parameters or input variables. Uncertain model parameters can be estimated by running a parameter identification on measured reference data or tuned in accordance with certain requirements by performing a parameter optimization. Systems with many degrees of freedom (DOF) and sophisticated actuation concepts typically result in very complex and high-dimensional multibody systems models [1]. High dimensionality accompanied by strong nonlinearities result in long evaluation times, large feasible regions and nonlinear constraints that impede the numerical solution of related optimization problems [9]. In order to handle this type of models, capable modeling approaches as well as efficient computational algorithms are necessary.

Existing software packages for modeling and simulating multibody systems can be divided into numerical and symbolic formalisms. Numerical software packages, e.g., MBDYN [10], RBDL[1], MOBY[2], apply efficient and flexible numerical subroutines to compute simulation results by exploiting structural properties of multibody systems. Many of these software packages are freely available and published under an open-source license, but are limited to kinematics and dynamics simulations. Symbolic software packages, e.g. ROBOTRAN [11], NEWEUL-M^2 [12], DRAKE[3], generate symbolic equations that are potentially very fast and powerful and allow further modification and processing like the computation of derivatives. Most of these software packages can be used free of charge, but are based on commercial algebra symbolic engines, e.g. MATLAB (MathWorks, USA), MAPLE (Waterloo Maple, Canada).

In this paper, the efficient and modular numerical multibody systems library MBSLIB [13] for kinematics and dynamics simulation, optimization and sensitivity analysis is presented. The library is written in *C++* and provides interfaces to external software packages and common model descriptions. A small footprint of the core library allows the application on various systems, while a modular software architecture and provided interfaces enable an easy extension. The main features of MBSLIB comprise

- an intuitive and straightforward modeling interface,
- a modular software architecture,
- efficient computational algorithms for forward, inverse and hybrid dynamics as well as forward kinematics,
- automatic differentiation with respect to system states, control variables or model parameters,

Manuscript received: August, 31, 2015; Revised December, 12, 2015; Accepted January, 7, 2016.

This paper was recommended for publication by Editor Kevin Lynch upon evaluation of the Associate Editor and Reviewers' comments. The work was partially funded by the German Research Foundation DFG (STR533/8-1).

The authors are with the Simulation, Systems Optimization and Robotics Group (SIM), Department of Computer Science, Technische Universität Darmstadt, Germany. `wojtusch | kunz | stryk@sim.tu-darmstadt.de`

Digital Object Identifier (DOI): see top of this page.

[1]http://rbdl.bitbucket.org

[2]http://physsim.sourceforge.net

[3]http://drake.mit.edu

- an interface for the direct collocation method DIRCOL [14] to solve optimal control problems,
- support for DH [15] and URDF[4] model descriptions.

The computation of derivatives with respect to system states, control variables or model parameters based on automatic differentiation is a prominent and expedient feature of MBSLIB. In combination with gradient-based optimization methods, these derivatives can be used to facilitate the numerical solution of optimization problems like optimal control or parameter estimation and optimization significantly. The derivatives also represent the sensitivities of the simulation outputs with respect to the specified independent variables and can be applied in differential sensitivity analysis. These features make MBSLIB a powerful, flexible and lightweight modeling, simulation and optimization library for many technical and biomechanical applications. Currently, MBSLIB is focused on tree-like multibody systems. The source code of the library is open and can be obtained free of charge from the MBSLIB website. Details on the modeling structure, modeling elements and implemented algorithms are given in Section II. A numerical validation of the included algorithms to compute kinematics, dynamics and derivatives is described in Section III. Section IV presents two application examples from robotics and biomechanics and a computation time analysis of the implemented algorithms. A concluding discussion is given in Section V.

II. MODELING AND SIMULATION

MBSLIB is based on the minimal coordinate formulation with relative coordinates [16]. Tree-structured multibody systems are modeled in a hierarchical model tree that consists of predefined or custom modeling elements [13]. The predefined modeling elements are fixed or free base, fixed translation or rotation, prismatic or revolute joint, rigid link, fork and endpoint. Examples for custom elements are links with specific geometries or multi-dimensional joints. Starting with a fixed or free base that forms the root node with a single branch, the required modeling elements are added sequentially to the model tree. A fork introduces a new branch. For a valid model tree, all branches need to be terminated by endpoints that represent leaf nodes. Forces and torques can be applied in joints and endpoints. Joints allow to set internal force or torque as well as position, velocity and acceleration directly. In addition, predefined or custom generators enable to apply external forces or torques on joints or endpoints as functions of system states or control variables. The predefined generators comprise a linear spring, an active linear spring and damper drive and a passive linear spring and damper drive. Examples for custom generators are nonlinear springs and dampers or muscle-tendon units. Generators such as linear springs can act on two or more endpoints and are modeled as polygon lines which connect the affected endpoints. The generated forces act along the polygon connection lines and are transformed into equivalent joint forces or torques. Figure 1 presents a partial class diagram of MBSLIB covering the described modeling element and generator classes and showing the fundamental software architecture.

The dynamic behavior of a multibody system is described by the motion equation

$$\boldsymbol{\tau} = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \boldsymbol{G}(\boldsymbol{q}) + \boldsymbol{F}_{ext} \qquad (1)$$

with the joint trajectories $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, $\ddot{\boldsymbol{q}}$, joint forces or torques $\boldsymbol{\tau}$, mass matrix $\boldsymbol{M}(\boldsymbol{q})$, Coriolis and centrifugal vector $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$, gravitation vector $\boldsymbol{G}(\boldsymbol{q})$ as well as joint forces or torques vector $\boldsymbol{F}_{ext}$ resulting from external forces or torques. Solving the motion equation 1 for the joint acceleration $\ddot{\boldsymbol{q}}$ is known as forward dynamics simulation. MBSLIB offers two established and efficient algorithms for solving forward dynamics problems [13]. The first method is the composite-rigid-body algorithm (CRBA) [17], [18] that computes the mass matrix $\boldsymbol{M}(\boldsymbol{q})$ first and solves a system of linear equations subsequently. This inertia matrix method has a computational complexity of $\mathcal{O}(n^3)$ for a multibody system with n joints. The second method is the articulated-body algorithm (ABA) [18], [19] that applies a recursive approach with three passes over the model tree. This propagation method has a computational complexity of $\mathcal{O}(n)$. Nevertheless, the ABA has a longer runtime than the CRBA for systems with less than nine joints [18]. In MBSLIB, both forward dynamics algorithms can be exchanged transparently in order to use the better performing method for the considered multibody system. Joint positions $\boldsymbol{q}$ and velocities $\dot{\boldsymbol{q}}$ can be determined by time integrating a series of computed joint accelerations $\ddot{\boldsymbol{q}}$. The time integration can be performed with any numerical integration software, e.g. ODEINT[5]. For testing purposes, an implementation of the explicit Euler algorithm is included in MBSLIB. Solving the motion equation 1 for the joint torque or force $\boldsymbol{\tau}$ is

[4]http://wiki.ros.org/urdf

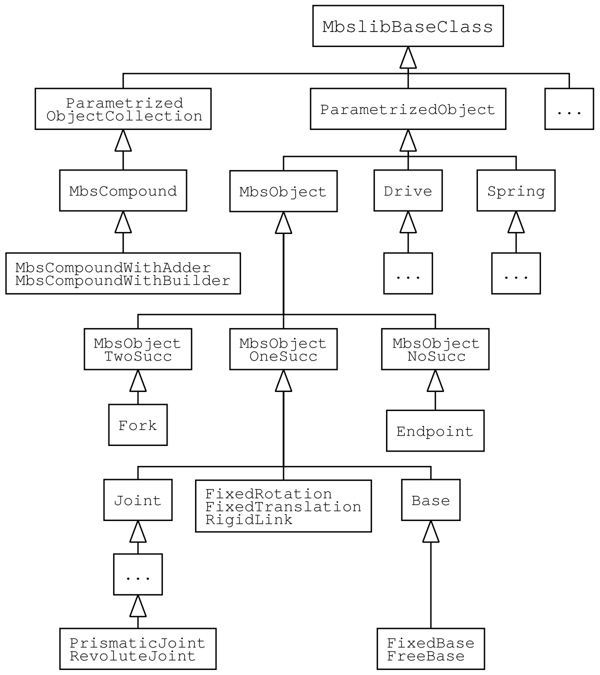


Figure 1: Partial class diagram of MBSLIB covering predefined modeling element, generator and compound classes.

[5]http://www.odeint.com

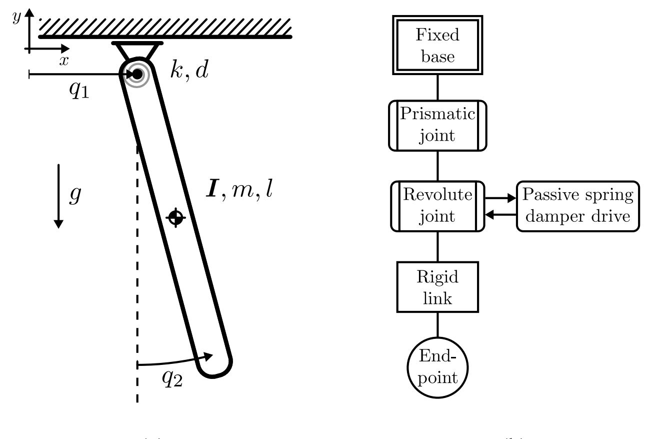


Figure 2: Schematic diagram (a) and model tree (b) of the pendulum on trolley.

known as inverse dynamics simulation. MBSLIB provides an implementation of the recursive Newton-Euler algorithm (RNEA) [20] with a computation complexity of $\mathcal{O}(n)$ for computing inverse dynamics. A mixed computation of forward and inverse dynamics, also referred to as hybrid dynamics, where each joint has an assigned joint acceleration $\ddot{\boldsymbol{q}}$ or joint torque or force $\boldsymbol{\tau}$ and the unknown joint variables are calculated accordingly, is supported in MBSLIB by providing the hybrid articulated-body algorithm (ABA-hybrid) [18].

For the computation of derivatives with respect to system states, control variables and model parameters, MBSLIB integrates the automatic differentiation package ADOL-C [21] in tape mode. Recent studies [22] showed that ADOL-C is well suited for multibody systems simulations and provides accurate results. ADOL-C works by operator overloading and introduces a new floating point data type that is used to record a tape of performed computations. By a subsequent interpretation of the recorded tape, desired derivatives can be obtained without truncation errors but with some run-time and memory overhead. In combination with ADOL-C, MBSLIB allows to compute explicit differentiation directly. Implicit differentiation like derivatives depending on joint angles computed from forward dynamics simulation requires to consider additional computation steps like time integration in the determination of derivatives by ADOL-C. Optionally, MBSLIB can be compiled with or without ADOL-C integration. To solve optimal control problems, MBSLIB provides an interface to the direct collocation method DIRCOL. An adapter class easily allows to create, configure and carry out model based optimizations with DIRCOL using the dynamics of any multibody systems model. Features of MBSLIB can be utilized in MATLAB (Mathworks, USA) by using the MATLAB interface to call *C* shared libraries in combination with a user-defined *C* library to invoke MBSLIB methods.

In order to provide an intuitive and straightforward modeling interface, the main functionalities of MBSLIB are encapsulated in the class `MbsCompoundWithBuilder`. The class listed in the partial class diagram in Figure 1 provides methods to setup a multibody systems model, set and read joint states, read modeling element reference frames and apply kinematics and dynamics simulations. A simple example system given by a compound pendulum with a linear rotary stiffness and damper mounted on a frictionless, massless trolley shown in Figure 2a is used to demonstrate the modeling procedure. The system consists of a fixed base, a prismatic joint q_1 modeling the trolley as well as a revolute joint q_2 and a rigid link modeling the pendulum with inertia I, mass m and length l. The linear stiffness and damper k, d are modeled by a passive spring damper drive that is connected to joint q_2. Gravitation acts in negative y direction of the global reference frame. Figure 2b illustrates the model tree of the system with the stated modeling elements as well as a terminating endpoint. A possible implementation of the system using MBSLIB is given in Listing 1. The first block creates a `MbsCompoundWithBuilder` object, sets gravitation and builds the multibody systems model by calling the corresponding adding methods. In the second block, the passive spring damper drive is set up and connected to the revolute joint. Initial joint positions and velocities are set in the third block. The last block evaluates the included drive and applies a forward dynamics simulation step by using the CRBA. For reasons of clarity and comprehensibility, the specific method parameters are left out in the given listing.

III. VALIDATION

Numerical validation allows to assess computational algorithms and is an important test for a correct implementation. Exemplary, the forward kinematics and forward and inverse dynamics algorithms as well as the automatic differentiation implemented in MBSLIB are validated by comparing the simulation results of a simple and low-dimensional system with the analytical solution and the simulation results of a

Listing 1: MBSLIB setup code of the pendulum on trolley.

```
#include <mbslib/mbslib.h>

using namespace mbslib;

// Set up pendulum on trolley
MbsCompoundWithBuilder mbs;
mbs.setGravitation(...);
mbs.addFixedBase();
Joint1DOF* q1 = mbs.addPrismaticJoint(...);
Joint1DOF* q2 = mbs.addRevoluteJoint(...);
mbs.addRigidLink(...);
mbs.addEndpoint();

// Set up passive spring and damper
PassiveSpringDamperDrive* psdd;
psdd = new PassiveSpringDamperDrive(*q2, ...);
mbs.addDrive(psdd);

// Set joint states
q1->setJointPosition(-0.5);
q1->setJointVelocity(0);
q2->setJointPosition(0.9 * M_PI);
q2->setJointVelocity(0);

// Apply forward dynamics
mbs.doForwardDrives();
mbs.doCrba();
```

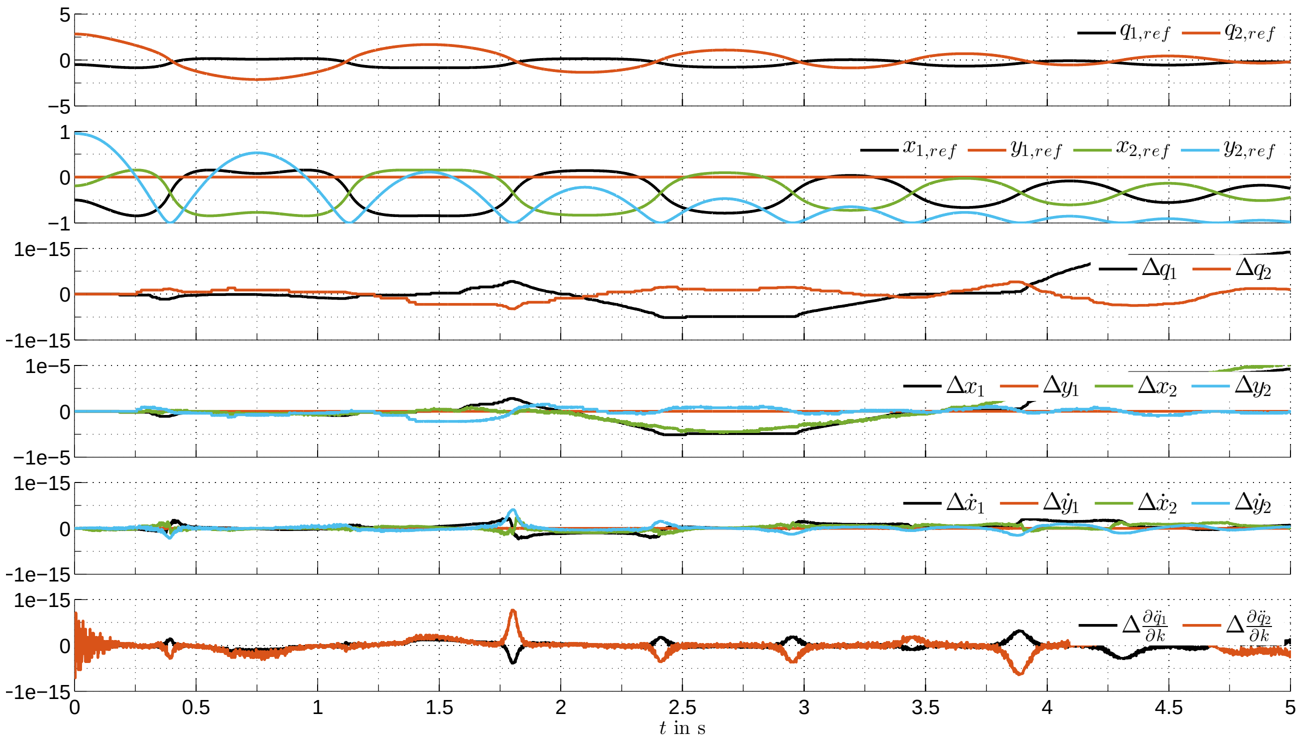


Figure 3: Simulation results of the pendulum on trolley given in SI units.

more complex and high-dimensional system with a given reference implementation.

A. Dynamics simulation of a pendulum on trolley

The first system is the pendulum on trolley described in Section II. The motion equation can be derived analytically by applying the Euler-Lagrange equation. The kinetic energy K, potential energy U and dissipative energy D are

$$\begin{aligned} K =&\frac{m}{2}\left(\frac{l}{2}\cos(q_2)\dot{q}_2+\dot{q}_1\right)^2+\frac{m}{2}\left(\frac{l}{2}\sin(q_2)\dot{q}_2\right)^2\\ &+\frac{1}{24}ml^2\dot{q}_2^2,\\ U =&\frac{1}{2}kq_2^2-\frac{1}{2}l\cos(q_2)mg, \quad D=\frac{1}{2}d\dot{q}_2^2. \end{aligned}$$

With the Lagrangian $L = K - U$ and the Euler-Lagrange equation $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} + \frac{\partial D}{\partial \dot{q}_i} = \tau_i,\ i = 1,\ldots,n$ the motion equations for q_1 and q_2 are given by

$$\begin{aligned} \ddot{q}_1 =&(2ml^2\sin(q_2)\dot{q}_2^2+6d\cos(q_2)\dot{q}_2+6k\cos(q_2)q_2\\ &+3gml\cos(q_2)\sin(q_2))/(4ml-3ml\cos(q_2)^2),\\ \ddot{q}_2 =&(3ml^2\cos(q_2)\sin(q_2)\dot{q}_2^2+12d\dot{q}_2+12kq_2\\ &+6gml\sin(q_2))/(3ml^2\cos(q_2)^2-4ml^2). \end{aligned}$$

The kinematic relation between the joint positions $\boldsymbol{q}$ and the Cartesian positions $\boldsymbol{x}$, $\boldsymbol{y}$ of the joint locations is specified by

$$\begin{aligned} x_1 =&q_1, & y_1 =&0,\\ x_2 =&q_1+l\sin(q_2), & y_2 =&-l\cos(q_2). \end{aligned}$$

For the validation, the system parameters are set to $l = 1.0\,\mathrm{m}$, $m = 5.0\,\mathrm{kg}$, $k = 10.0\,\mathrm{N\,m\,rad^{-1}}$, $d = 0.6\,\mathrm{N\,m\,s\,rad^{-1}}$, $g = 9.81\,\mathrm{m\,s^{-2}}$ and a forward dynamics simulation with the initial values $q_1 = -0.5\,\mathrm{m}$, $q_2 = 0.9\pi\,\mathrm{rad}$, $\dot{q}_1 = \dot{q}_2 = 0$ is carried out. Time integration is performed by applying the Runge-Kutta 4 algorithm implemented in ODEINT with a fixed step size of $h = 1.0\,\mathrm{ms}$. Figure 3 shows the motion of the pendulum on trolley and the comparison between the simulation results obtained by MBSLIB and the reference values given by the analytical solution. The absolute error of the joint trajectories $\Delta\boldsymbol{q}$, Cartesian positions $\Delta\boldsymbol{x}$, $\Delta\boldsymbol{y}$ and Cartesian velocities $\Delta\dot{\boldsymbol{x}}$, $\Delta\dot{\boldsymbol{y}}$ shown in the middle plots of Figure 3 that is computed by subtracting the reference values from the simulation results is very small and originates from finite numerical precision.

In addition to the forward dynamics simulation, the sensitivity of the joint accelerations $\ddot{\boldsymbol{q}}$ with respect to the stiffness k was determined. The analytical sensitivities for $\ddot{q}_1$ and $\ddot{q}_2$ applying explicit differentiation are given by

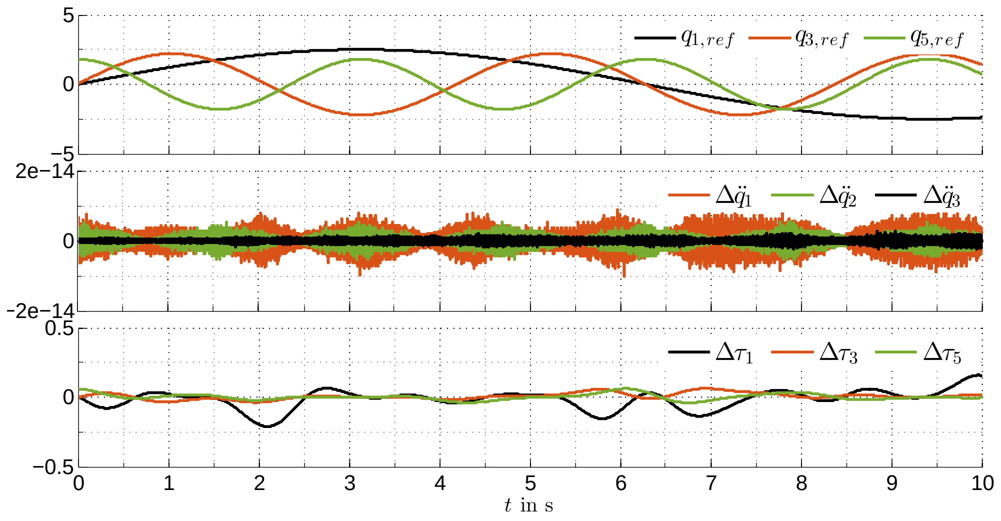
$$\begin{aligned} \frac{\partial\ddot{q}_1}{\partial k} =&-\frac{6q_2\cos(q_2)}{ml(3\cos(q_2)^2-4)},\\ \frac{\partial\ddot{q}_2}{\partial k} =&\frac{12q_2}{ml^2(3\cos(q_2)^2-4)}. \end{aligned}$$

In MBSLIB, the numerical sensitivities are computed by automatic differentiation. The lowest plot in Figure 3 shows the absolute error of the sensitivities $\Delta\frac{\partial\boldsymbol{q}}{\partial k}$. Likewise, the difference is limited to very small numerical deviations.

B. Dynamics simulation of a robot arm

The second system is the industrial robot arm MANUTEC R3 (manutec, Germany) shown in Figure 4a that consists of six rigid links and six revolute joints. A validated reference

(a)



(b)

Figure 4: Picture (a) and simulation results (b) of the robot arm MANUTEC R3 (manutec, Germany) given in SI units.

implementation [3] written in *Fortran* provides functions to evaluate forward and inverse dynamics. These functions also consider rotor dynamics that include moments of deviation resulting from the interaction between joint and actuator axes. A predefined modeling element that allows to regard theses effects does not yet exist in MBSLIB. The additional terms required for the extended mass matrix $\boldsymbol{M}(\boldsymbol{q})$ are obtained from a documented modeling approach with rotor dynamics [1] and added to the implementation of the CRBA. Figure 4b presents the absolute error of the joint accelerations $\Delta\ddot{\boldsymbol{q}}$ and joint torques $\Delta\boldsymbol{\tau}$ computed by subtracting the reference values from the simulation results for three exemplary joints generated by sinusoidal reference joint trajectories $\boldsymbol{q}_{ref}$ shown in the upper plot. The joint torques $\boldsymbol{\tau}$ are computed by inverse dynamics simulation, while the joint accelerations $\ddot{\boldsymbol{q}}$ result from forward dynamics simulation with the previously determined joint torques $\boldsymbol{\tau}$. The simulation results fit well, but the absolute error of the joint torques $\boldsymbol{\tau}$ exceeds expected numerical deviations. The absolute error varies between $-0.21\,\mathrm{Nm}$ and $0.16\,\mathrm{Nm}$ for the joint torque τ_1 that reaches values between $-64.7\,\mathrm{Nm}$ and $59.7\,\mathrm{Nm}$. Presumably, these variations are caused by small differences in the applied modeling approach and the implementation of the rotor dynamics.

IV. EXAMPLES

Two application examples from robotics and biomechanics as well as an analysis of the computation time required by the implemented algorithms are presented to demonstrate the practical relevance and applicability of MBSLIB. The source code of both examples is available on the MBSLIB website.

A. Parameter estimation of an elastic robot arm

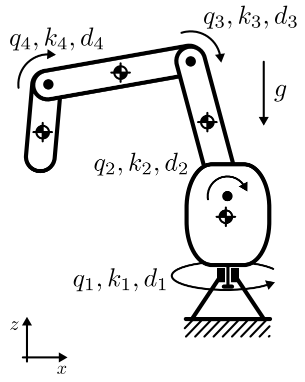
The first example is based on the elastic robot arm BIOROB X4 (Bionics Robotics, Germany) shown in Figure 5a. The multibody systems model of the robot arm illustrated in Figure 5b consists of four rigid links and four revolute joints. The robot arm applies series elastic tendon-driven actuators that are modeled as ideal rotary drives in series with linear rotary stiffnesses k_i and linear rotary dampers d_i. It is assumed that the rotary stiffnesses k_i and rotary dampers d_i are unknown and should be identified by parameter estimation from a measured joint acceleration trajectory $\ddot{\boldsymbol{q}}_m$ with a duration of 20.0 s sampled at 100 Hz and resulting from known sinusoidal series elastic actuator motions.

The parameter estimation is a non-linear least-squares problem with the parameter vector $\boldsymbol{p} = [\boldsymbol{k}, \boldsymbol{d}]^T$ and the residual vector $\boldsymbol{r}(\boldsymbol{p}) = \ddot{\boldsymbol{q}}(\boldsymbol{p}) - \ddot{\boldsymbol{q}}_m$. For solving this least-squares problem, the iterative Levenberg-Marquardt algorithm [9] is applied. The algorithm uses a damped form of the linear normal equation given by

$$\left(\boldsymbol{J}(\boldsymbol{p}^{(t)})\boldsymbol{J}^T(\boldsymbol{p}^{(t)}) + \mu^{(t)}\boldsymbol{E}\right)\boldsymbol{d}^{(t)} = -\boldsymbol{J}(\boldsymbol{p}^{(t)})\boldsymbol{r}(\boldsymbol{p}^{(t)})$$

to find the shift vector $\boldsymbol{d}^{(t)} = \boldsymbol{p}^{(t+1)} - \boldsymbol{p}^{(t)}$, where $\boldsymbol{J}(\boldsymbol{p}^{(t)}) =$

(a)



(b)

Figure 5: Picture (a) and schematic diagram (b) of the elastic robot arm BIOROB X4 (Bionics Robotics, Germany).

Table I: Results of the parameter estimation of the elastic robot arm given in SI units.

	Reference values	Start values	Ideal measurements	Noisy measurements
k_1	45.0	20.0	45.0	44.754
k_2	40.0	20.0	40.0	40.401
k_3	25.0	20.0	25.0	24.723
k_4	15.0	20.0	15.0	14.984
d_1	0.22	0.1	0.22	0.2237
d_2	0.18	0.1	0.18	0.1905
d_3	0.12	0.1	0.12	0.1140
d_4	0.14	0.1	0.14	0.1380

$\frac{\partial \boldsymbol{r}(\boldsymbol{p}^{(t)})}{\partial \boldsymbol{p}^{(t)}}$ is the Jacobian matrix, $\mu^{(t)}$ is the damping factor and $\boldsymbol{E}$ is the identity matrix. The joint acceleration trajectories $\ddot{\boldsymbol{q}}$ are computed with the CRBA and the Jacobian matrix $\boldsymbol{J}(\boldsymbol{p}^{(t)})$ is obtained by using automatic differentiation.

Table I summarizes the reference values, start values and parameter estimation results for ideal measurements without noise and noisy measurements including additive Gaussian white noise with $3\,\sigma = 0.1\,\max(\ddot{\boldsymbol{q}}_m)$. The results show that the reference values of the rotary stiffnesses k_i and rotary dampers d_i can be identified very well from the noisy measurements. The perfect identification result from the ideal measurements is hardly surprising, but validates the implementation of the Levenberg-Marquardt algorithm.

B. Joint trajectory estimation of human locomotion

The second example addresses joint trajectory estimation from measured motion capture data of human locomotion. The open HUMOD Database [8] provides biomechanical measurement data including motion capture data, ground reaction forces and muscle activities for eight different motion tasks. A three-dimensional motion capture system and a set of thirty-six reflective markers were used to record the motion. Soft tissue artifacts and other measurement errors may result in length variations of rigid assumed body segments that impair the estimation of joint trajectories. For fast running motions, this variation can reach up to 8 % in the thigh segment relative to an averaged reference length [8]. In order to compensate these influences, an extended Kalman smoother in combination with a subject-specific multibody systems model with twenty-eight DOF is used to estimate the joint trajectories [23]. Figure 6 illustrates the applied human body model with thirteen rigid links and twelve joints as well as the reflective marker set.

For estimating the joint trajectories $\boldsymbol{x}$ including joint positions $\boldsymbol{q}$, joint velocities $\dot{\boldsymbol{q}}$, joint accelerations $\ddot{\boldsymbol{q}}$ and joint jerk $\dddot{\boldsymbol{q}}$, the extended Kalman smoother combines noisy measurements of the reflective markers $\boldsymbol{z}$ with prior system knowledge and minimizes the estimation error statistically. The prior system knowledge is given by a process model $\boldsymbol{f}(\boldsymbol{x})$ that describes the expected time evolution of the joint trajectories $\boldsymbol{x}$ and a measurement model $\boldsymbol{h}(\boldsymbol{x})$ that characterizes the relation between the joint trajectories $\boldsymbol{x}$ and noisy measurements $\boldsymbol{z}$. The linear process model $\boldsymbol{f}(\boldsymbol{x})$ is obtained by assuming a constant joint jerk $\dddot{\boldsymbol{q}}$. The non-linear measurement model $\boldsymbol{h}(\boldsymbol{x})$ is given by the forward kinematics simulation of the multibody systems model. The extended Kalman smoother applies three successive update steps given by the recursive equations

$$\hat{\boldsymbol{x}}^{(t)} = \boldsymbol{f}(\boldsymbol{x}^{(t-1)}), \tag{2}$$

$$\tilde{\boldsymbol{x}}^{(t)} = \hat{\boldsymbol{x}}^{(t)} + \boldsymbol{K}^{(t)}\left(\boldsymbol{z}^{(t)} - \boldsymbol{h}(\hat{\boldsymbol{x}}^{(t)})\right), \tag{3}$$

$$\boldsymbol{x}^{(t)} = \tilde{\boldsymbol{x}}^{(t)} + \boldsymbol{J}^{(t)}\left(\boldsymbol{x}^{(t+1)} - \boldsymbol{f}(\tilde{\boldsymbol{x}}^{(t)})\right) \tag{4}$$

with the adaptive filter gain $\boldsymbol{K}^{(t)}$ and smoother gain $\boldsymbol{J}^{(t)}$. The first step in Equation 2 is a predictive time update using the process model $\boldsymbol{f}(\boldsymbol{x})$. In the second step in Equation 3, a forward recursion (filtering) estimates the joint trajectories $\boldsymbol{x}$ from the measurements $\boldsymbol{z}$. The third step in Equation 4 applies a backward recursion (smoothing) to further improve the joint trajectory estimates $\boldsymbol{x}$. For computing the adaptive smoother gain $\boldsymbol{J}^{(t)}$, the Jacobian matrix $\boldsymbol{H}(\boldsymbol{x}^{(t)}) = \frac{\partial \boldsymbol{h}(\boldsymbol{x}^{(t)})}{\partial \boldsymbol{x}^{(t)}}$ representing the derivative of the non-linear forward kinematics with respect to the joint trajectories $\boldsymbol{x}$ is required. MBSLIB is used to compute the forward kinematics $\boldsymbol{h}(\boldsymbol{x})$ as well as to obtain the Jacobian matrix $\boldsymbol{H}(\boldsymbol{x}^{(t)})$ by applying automatic differentiation. The resulting smoothed joint trajectories without segment length variations are provided in the HUMOD Database.

C. Analysis of computation time

The computation time of the implemented forward kinematics (KIN), forward dynamics (CRBA, ABA) and inverse dynamics (RNEA) algorithms with and without ADOL-C integration as well the interpretation of the tape created by ADOL-C to compute derivatives is evaluated by applying the models of the pendulum on trolley (A) described in Section III-A and the human body (B) described in Section IV-B.

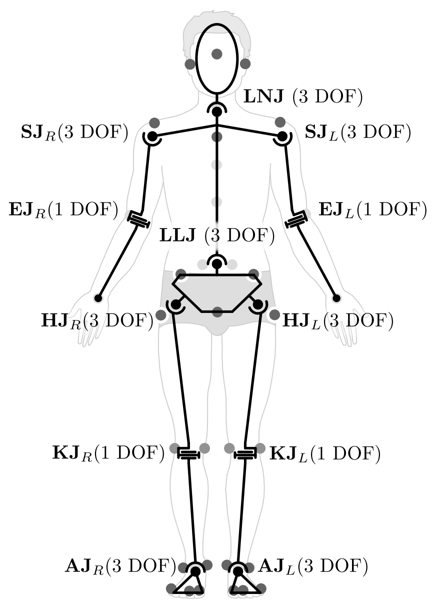


Figure 6: Schematic diagram of the human body model with twenty-eight DOF and thirty-six reflective markers placed anterior (dark grey) and posterior (light grey).

Table II: Results of the computation time analysis for one algorithm run or derivative computation given in microseconds.

	KIN	CRBA	ABA	RNEA
A: run without ADOL-C	0.01	1.03	1.04	0.01
A: run with ADOL-C	13.6	83.8	189.9	47.3
A: derivative from tape	102.3	135.6	200.4	169.5
B: run without ADOL-C	16.6	68.6	44.4	22.3
B: run with ADOL-C	305.8	4393.4	3584.0	967.2
B: derivative from tape	1197.1	3738.7	3226.8	1475.2

A computer system with i7-4810MQ processor (Intel, USA) at a constant clock rate of 2.8 GHz and Ubuntu 14.04 is used for running the analysis. The computation times for one run of each algorithm and one derivative computation are determined by averaging one thousand runs with random, evenly distributed input variables $\boldsymbol{q} \in [-\frac{\pi}{2}, \frac{\pi}{2}]$, $\dot{\boldsymbol{q}} \in [-\frac{10\pi}{2}, \frac{10\pi}{2}]$, $\ddot{\boldsymbol{q}} \in [-\frac{100\pi}{2}, \frac{100\pi}{2}]$ and $\boldsymbol{\tau} \in [-100, 100]$ given in SI units. For the derivative computation, arbitrary derivatives $\frac{\partial q_i}{\partial l_i}$ in forward kinematics, $\frac{\partial \ddot{q}_i}{\partial m_i}$ in forward dynamics and $\frac{\partial \tau_i}{\partial m_i}$ in inverse dynamics are calculated. Table II lists the averaged computation times for all algorithms and the derivative computation. The results show that all algorithms are very efficient without ADOL-C integration. For model A with two DOF, the CRBA performs slightly better than the ABA, while the ABA runs faster than the CRBA for model B with twenty-eight DOF. This result illustrates the advantage of the ABA in systems with nine or more joints as stated in Section II. With ADOL-C integration, the computation times of all algorithms increase significantly, which is also reflected in the computation times of the derivatives. The overall performance can be improved by optimizing the ADOL-C integration for example by reusing specific ADOL-C objects.

V. Conclusion

MBSlib is a powerful, flexible and lightweight modeling, simulation and optimization library suitable for many technical and biomechanical applications. It features an intuitive modeling interface and modular software architecture in combination with efficient computational algorithms for forward and inverse dynamics as well as forward kinematics simulations. The possibility to compute derivatives with respect to system states, control variables or model parameters is a prominent and expedient feature and allows to facilitate the numerical solution of optimization problems significantly. The computational validity of the implemented algorithms and automatic differentiation approach is demonstrated on a simple and low-dimensional system with analytical solution and a more complex and medium-dimensional system with reference implementation. For showing the practical relevance and applicability of MBSlib, two application examples from robotics and biomechanics are presented. The intuitive modeling interface, modular architecture and provided interfaces enable an easy extension or integration into other projects.

The source code of MBSlib is open and published under the Lesser General Public License (LGPL v3). It can be obtained free of charge from the MBSlib website:

http://www.sim.informatik.tu-darmstadt.de/mbslib

The simple and free availability of MBSlib allows interested users to utilize, share and improve the library. This is supposed to enhance functionality, quality as well as usability. For future releases, support of closed-loop multibody systems, methods for contact handling and further optimization of the ADOL-C integration are planned.

References

[1] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*, 1st ed. Springer, 2008.

[2] D. A. Winter, *Biomechanics and Motor Control of Human Movement*, 4th ed. John Wiley & Sons, 2009.

[3] M. Otter and S. Türk, "The DFVLR Models 1 and 2 of the Manutec r3 Robot," Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt, Köln, Germany, Tech. Rep. 88-13, 1988.

[4] T. Lens and O. von Stryk, "Investigation of safety in human-robot-interaction for a series elastic, tendon-driven robot arm," in *IEEE/RSJ IROS*, 2012, pp. 4309–4314.

[5] K. Kaneko, K. Harada, F. Kanehiro, G. Miyamori, and K. Akachi, "Humanoid robot HRP-3," in *IEEE/RSJ IROS*, 2008, pp. 2471–2478.

[6] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson, "Optimization-based Full Body Control for the DARPA Robotics Challenge," *J. Field Rob.*, vol. 32, no. 2, p. 293–312, 2015.

[7] E. M. Arnold, S. R. Ward, R. L. Lieber, and S. L. Delp, "A Model of the Lower Limb for Analysis of Human Movement," *Ann. Biomed. Eng.*, vol. 38, no. 2, pp. 269–279, 2009.

[8] J. Wojtusch and O. von Stryk, "HuMoD - A versatile and open database for the investigation, modeling and simulation of human motion dynamics on actuation level," in *IEEE/RAS Humanoids*, Seoul, 2015.

[9] J. Nocedal and S. J. Wright, *Numerical optimization*, 2nd ed. Springer, 2006.

[10] P. Masarati, M. Morandini, and P. Mantegazza, "An Efficient Formulation for General-Purpose Multibody/Multiphysics Analysis," *J. Comput. Nonlinear Dyn.*, vol. 9, no. 4, pp. 1–9, 2014.

[11] N. Docquier, A. Poncelet, and P. Fisette, "ROBOTRAN: A powerful symbolic generator of multibody models," *Mech. Sci.*, vol. 4, no. 1, pp. 199–219, 2013.

[12] T. Kurz, P. Eberhard, C. Henninger, and W. Schiehlen, "From Neweul to Neweul-M^2: Symbolical equations of motion for multibody system analysis and synthesis," *Multibody Sys. Dyn.*, vol. 24, no. 1, pp. 25–41, 2010.

[13] M. Friedmann, J. Wojtusch, and O. von Stryk, "A modular and efficient approach to computational modeling and sensitivity analysis of robot and human motion dynamics," *Appl. Math. Mech.*, vol. 12, no. 1, pp. 85–86, 2012.

[14] O. von Stryk, "User's Guide for DIRCOL - A Direct Collocation Method for the Numerical Solution of Optimal Control Problems," Technische Universität Darmstadt, Darmstadt, Germany, Tech. Rep. 2.1, 2002.

[15] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *J. Appl. Mech.*, vol. 77, no. 23, pp. 215–221, 1955.

[16] R. von Schwerin, *MultiBody System SIMulation*, 1st ed. Springer, 1999.

[17] M. W. Walker and D. E. Orin, "Efficient Dynamic Computer Simulation of Robotic Mechanisms," *J. Dyn. Syst. Meas. Contr.*, vol. 104, no. 3, p. 205, 1982.

[18] R. Featherstone, *Rigid Body Dynamics Algorithms*, 1st ed. Springer, 2008.

[19] ——, "The Calculation of Robot Dynamics Using Articulated-Body Inertias," *Int. J. Robot. Res.*, vol. 2, no. 1, pp. 13–30, 1983.

[20] J. Y. S. Luh, M. W. Walker, and R. P. C. Paul, "On-Line Computational Scheme for Mechanical Manipulators," *J. Dyn. Syst. Meas. Contr.*, vol. 102, no. 2, p. 69, 1980.

[21] A. Walther and A. Griewank, "Getting started with ADOL-C," in *Combinatorial Scientific Computing*, 1st ed. Chapman-Hall CRC Computational Science, 2012, ch. 7, pp. 181–202.

[22] A. Callejo, S. H. K. Narayanan, J. G. de Jalón, and B. Norris, "Performance of automatic differentiation tools in the dynamic simulation of multibody systems," *Adv. Eng. Software*, vol. 73, no. 7, pp. 35–44, 2014.

[23] F. De Groote, T. De Laet, I. Jonkers, and J. De Schutter, "Kalman smoothing improves the estimation of joint kinematics and kinetics in marker-based human gait analysis," *J. Biomech.*, vol. 41, no. 16, pp. 3390–3398, 2008.