A Maple Toolchain for Rigid Body Dynamics of Serial, Hybrid and Parallel Robots

Moritz Schappler, Tim-David Job, and Tobias Ortmaier

Institute of Mechatronic Systems, Leibniz University Hannover, Germany
{moritz.schappler,tim-david.job,tobias.ortmaier}@imes.uni-hannover.de

Abstract. A new Maple toolchain for generating rigid body dynamics in symbolic form for robot manipulators is presented. The peculiarity compared to existing tools lies in the framework of Bash scripts controlling the full workflow of the toolchain with a high degree of automation. The optimized Matlab code generated by Maple is automatically converted to function files with proper documentation and input assertions. This renders manual post-processing of the results unnecessarily. The focus of the paper is on the implemented unit-testing framework according to the method of test-driven development. By providing the test framework together with the generated code in a stand-alone version, a good test coverage and a good software quality can be achieved. The results of the open source project provide a basis for dynamics simulations for robot dimensional synthesis or in model-based control of robot manipulators in research or in industrial context. The general software approach can be applied to other fields where theoretical models are derived with Maple.

Keywords: Rigid body dynamics · Robotics · Symbolic code · Toolchain · Test-driven development · Maple computer algebra system

1 Introduction and State of the Art

Using a symbolic rather than numeric implementation of dynamics models for robots is highly beneficial regarding the computational efficiency [9]. Some aspects of the models can only be obtained in a useful way via symbolic derivation, such as the identification model [12]. Using models in simulations for a comparison of different robots requires an automatic, general and efficient approach.

To be able to find the robot that is suited best for a given task, first a set of robot kinematics has to be created, which is the outcome of the *structural synthesis*. The structural synthesis of serial robots can be performed using screw theory [17], Denavit-Hartenberg parameters [23] or variants thereof, such as the traveling coordinate system method [10]. The synthesis of serial chains is mostly discussed in literature in the context of parallel robots, which contain several serial kinematic leg chains connected to a moving platform. Leg chains are generated with the virtual-chain approach and screw theory [15] or using the theory of linear transformations and evolutionary morphology [11]. Following [19, p.25] (for parallel robots), for the assessment of the robot's performance, the dimensions of the design parameters (e.g. lengths of the links) are as important as

the kinematic structure itself (i.e. number, type and alignment of joints). This leads to the requirement of a *combined robot synthesis*, as proposed in [16] for parallel robots, to determine which robot structure is suited best for a given task. A *dimensional synthesis*, i.e. an optimization of the robot's dimensions, has to be performed for all possible structures. First investigations on the combined synthesis for serial robots [23] have shown that the approach is feasible in principle. However, the practicability of such an extensive optimization of hundreds of robots with several optimization parameters each and highly nonlinear models strongly depends on the implementation. The robot models and the objective and constraints function within the optimization problem need an efficient implementation to be able to generate substantiated results.

A simulation of the *robot dynamics model* (i.e. the relation of force and motion) has to be evaluated in each iteration of the aforementioned optimization. The rigid body dynamics for robots itself is a mathematical problem that can be considered solved in that context for serial [12], hybrid [13,26,9,24,6] and parallel robots [19,3,5,1]. For serial kinematic chains the Newton-Euler algorithm is mainly used in software dedicated for robot dynamics [13,14,24]. Hybrid robots, i.e. serial robots with additional closed kinematic loops, are mainly modeled based on the serial chain dynamics with additional variational principles to take closed loops into account [24] (D'Alembert, Jourdain). For parallel robots, different definitions of the system coordinates are possible based on these principles of energy equivalence [3,5,1].

There exist a variety of software toolchains for modeling dynamics equations for robot manipulators. A probably non-exhaustive list contains the symbolic tools Robotran [9,24,6], SYMORO [13], openSYMORO [14], MapleSim [31], Neweul-M^2 [18], the Peter Corke Matlab toolbox [4] and some open source toolboxes from single research projects, such as FloBaRoID [2], SymPyBotics (or SageRobotics [27]) and the dVRK Dynamic Model Identification Package [30]. Several numeric tools are available for simulating the inverse and forward dynamics of general multibody systems, which includes the robot manipulators in this work. Prominent examples are MSC Adams [20], Matlab Simscape Multibody (SimMechanics) [29], the Rigid Body Dynamics Library [8], based on Featherstone's theory [7], and Drake [28]. Some symbolic programs also provide the possibility for a numeric simulation of the systems, such as MapleSim [31].

These toolchains do not directly meet the requirements for creating a model database required for the combined synthesis. Extensions regarding batch-processing, unit-testing and post-processing are required, since most tools require user interaction, which is not feasible for hundreds of robots. A key method for ensuring software quality is systematic testing using unit testing frameworks, which is central to test-driven development. This is often disregarded in software for scientific projects [32]. Available toolchains presumably all give correct results, but it is not always possible to completely verify this by the end user. Open source tools, such as [27,14,2], typically come explicitly without warranty raising the need for additional validation of the results. Misinterpretation of interfaces of not well documented, but still error-free, software modules may

introduce errors in the further use within a bigger project. Creating a robot database with the claim to include every unique robot structure will generate all possible test cases and will raise all existing software bugs. Not using a proper testing environment therefore can put unnecessary risks on projects relying on the results of the software tools.

To encounter these issues for the case of robot dynamics for the proposed application, a new toolchain[1] was developed. It is based on Maple as symbolic engine and Bash scripts for an automation of the model generation process. This provides the flexibility to test the implementation of different algorithms, e.g. an efficient formulation for parallel robots [1] or an unconventional method for hybrid robot dynamics [25]. The contributions of this paper are

- a comparison of existing tools for the symbolic form of robot dynamics,
- elaborations on performing unit testing for parts of theoretical models at the example of robot dynamics,
- details on the implementation of the new toolchain, which may be used to structure similar programs in other fields,
- the application of the toolchain to a robot model database.[2]

The remainder of the paper is structured as follows. An overview of existing programs is given Sec. 2. Theoretical fundamentals of robot dynamics are summarized in Sec. 3 with a focus on how to perform unit testing. The structure of the toolchain is presented in Sec. 4. The robot database as application example is introduced in Sec. 5 and Sec. 6 concludes the paper.

2 Comparison of Existing Toolboxes for Robot Dynamics

As sketched in the previous section, several tools already exist for generating the rigid body dynamics of robots in symbolic form. An extensive comparison of the tools is given in Table 1. Some older software packages, e.g. referenced in [27] are left out of the comparison due to their presumed deprecation. Commercial software, such as Robotran and MapleSim, is available at a mature stage of development. Since OpenSymoro is publicly available, the necessity escapes to use Symoro+ with similar features. A variety of open source projects for robot manipulators is implemented in Python using the `sympy` library as computer algebra system (CAS) which helps avoiding licensing costs for software like Maple, Mathematica, MapleSim and Matlab. The drawback of open source tools is the dependency on single researchers supporting them, as can be seen by the status "unmaintained" of SymPyBotics or the GitHub list of issues of OpenSymoro.

Robotran, MapleSim and Neweul-M^2 allow the derivation of multibody dynamics of general mechanisms, which includes both tree-like and closed-loop systems and therefore all types of common robot manipulators. The Python tools mainly focus on robotic applications, e.g. only serial robots [27] or additionally robot kinematics with closed loops [14,30]. Some have very specific focus, such

[1] Available under free license at https://github.com/SchapplM/robsynth-modelgen.
[2] The database is available at https://github.com/SchapplM/robsynth-serroblib for serial robots, ...-serhybroblib for hybrid and ...-parroblib for parallel robots.

as humanoid robots [2] or dynamics model identification [30,2]. Parallel robots (PKM, parallel kinematic machines) require a specific modeling approach (see Sec. 3.3), which is not available in the open source tools, but can be obtained by MapleSim or Robotran. If a general closed-loop robot model not in minimal (platform) coordinates is used instead, this leads to a less efficient implementation, since coordinates of platform coupling joints remain in the equations.

Some multibody tools have graphical user interfaces that reduce the need of expert knowledge. For the batch creation of a robot database, this strength can become a weakness, if it is not possible to automatically generate the dynamics equations from a standardized description of the robot. A new toolchain was developed, partly to avoid dependencies on dedicated commercial tools (while allowing the dependency on a commercial CAS), partly due to the fact that most open source tools were not accessible at the begin of the work in 2015. Since an institutional license was available, the core tools of the proposed toolchain are Maple for the symbolics engine and Matlab for the model evaluation and simulations. This design decision distinguishes the proposed toolchain from the Python toolboxes which have no commercial dependencies.

Table 1. Comparison of different tools for symbolic robot dynamics (legend below)

Name	Ref.	Area (1)	License (2)	CAS (3)	IM (4)	FlB (5)	Year	UI (6)
Robotran	[6]	Multibody	Comm. (7)	MBS (8)	yes	yes	1990	GUI/CMD
Robotica	[22]	OL Rob.	OSS	Mathematica	no	no	1994	CMD+Vis.
Symoro+	[13]	OL/CL Rob.	Comm.	Mathematica	yes	no	1997	GUI
MapleSim	[31]	Multibody	Comm.	Maple	?	?	2000	GUI
Neweul-M^2	[18]	Multibody	Pr. (9)	Matlab	no	yes	2007	GUI/CMD
ParaDyn	[5]	OL/PKM	Pr. (10)	Maple	yes	no	2009	CMD
RVC toolbox	[4]	OL Rob.	OSS	Matlab	no	no	2012	CMD+Vis.
OpenSymoro	[14]	OL/CL Rob.	OSS	Python	yes	yes	2014	CMD+Vis.
SymPyBotics	[27]	OL Rob.	OSS	Python	yes	no	2014	CMD+Vis.
FloBaRoID	[2]	OL Rob.	OSS	Python	yes	yes	2016	CMD+Vis.
dVRK DMI	[30]	OL/CL Rob.	OSS	Python	yes	no	2019	CMD
Proposed		OL/CL/PKM	OSS	Maple	yes	yes	2019	CMD

Legend for Table 1 (referenced by round brackets in table headings and rows):
1: Area of application: multibody: general m.b. dynamics; OL Rob.: open loop robots; CL Rob.: closed-loop robots; PKM: parallel kinematic machines (parallel robots).
2: OSS: Open source software; comm.: commercial software; pr.: proprietary tool.
3: Additional license required for Mathematica, Matlab Symbolics Toolbox or Maple.
4: Identification model of the inverse dynamics (linear in the dynamics parameters).
5: Floating base model for the inverse dynamics (non-fixed base link with six DoF).
6: UI: User interface; GUI: graphical; CMD: command line; Vis.: visualisation of the results (but no visual interface for input).
7: Free for teaching and academic research.
8: Dedicated CAS for multibody systems (see [24]), accessed via web-based service.
9: Access to the software provided for project partners from industry and academia.
10: The tool was used for several projects from 2009 to 2017 at the author's institute.

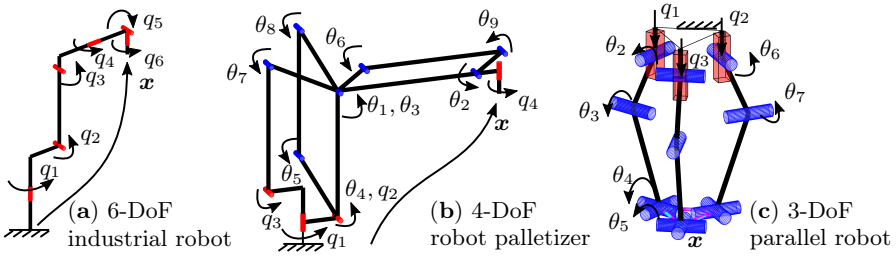


Fig. 1. Examples of different types of robots with annotation of coordinates $\boldsymbol{q}$, $\boldsymbol{\theta}$, $\boldsymbol{x}$. Cylinders and cuboids mark revolute and prismatic joints.

3 Robot Dynamics and Unit Testing Framework

The following section contains a high-level summary of the kinematic and dynamics models of the three types of robots implemented in the proposed toolchain. The modeling usually starts from a kinematic sketch of the robot, which may originate from an existing CAD model. Detailed derivations and explanations can be obtained from standard textbooks, such as [19,12,3,24], and the research papers that are referenced. The theory is the basis for the unit testing framework and therefore every part of the model (structured into numbered properties) is followed by an elaboration on how to perform a unit test on it. It is assumed that the model is derived symbolically but tests are performed numerically, since the output of the tool are functions which implement single terms of the model. A symbolic check for equality of two expressions is often not feasible, especially if the derivation is by different approaches. The structure of this section enables to follow the transfer from theory to test cases and allows an adaption of the approach to theoretical models from other fields. The theoretical framework is restricted to rigid body dynamics for three different types of robots, which each require a specific approach to derive efficient models. The three types of robots are sketched in Fig. 1. Serial robots (Fig. 1,a) are discussed in Sec. 3.1, serial-hybrid robots (Fig. 1,b) in Sec. 3.2 and parallel robots (Fig. 1,c) in Sec. 3.3.

3.1 Serial-Link Robots

Fundamentals for serial robots are taken from the standard textbook [12]. The derivation of the theory is structured according to the basic dynamics principles of Newton-Euler and Lagrange, which start with the relations of position and velocity (kinematics), over the definition of energy to forces (dynamics).

The following list of examples is not exhaustive for serial robots, but gives an impression on how to prove the validity of the results for all steps of the derivation of the kinematic and dynamics equations. The theoretical properties and test cases of the models are partly summarized in Fig. 2. For a good test coverage, each property block should have a dashed connection to a test case block.

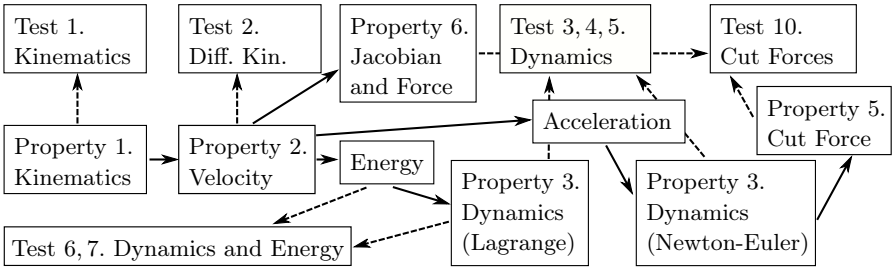


Fig. 2. Overview of the properties and tests for serial robots

Property 1. The end effector pose $\boldsymbol{x}$ can be calculated via $\boldsymbol{x}=\boldsymbol{f}(\boldsymbol{q})$ for a given vector of n joint coordinates $\boldsymbol{q}$. The *forward kinematics* is implemented using homogenous transformation matrices and the modified Denavit-Hartenberg parameters from [12] for a minimal-parameter representation of the joint transformations. The pose $\boldsymbol{x}$ is without loss of generality a vector of position and three Euler angles expressing orientation.

Test 1. The function $\boldsymbol{f}(\boldsymbol{q})$ can be validated graphically with plots, such as Fig. 1,a. A CAD model can facilitate this, if available, but this is not mandatory.

Property 2. The *velocity* of the end effector follows the linear relation $\boldsymbol{v}=\boldsymbol{J}_{\mathrm{g}}(\boldsymbol{q})\dot{\boldsymbol{q}}$ and $\dot{\boldsymbol{x}}=\boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}$, which represents the *differential kinematics*. The velocity $\boldsymbol{v}$ contains linear velocity and angular velocity, while $\dot{\boldsymbol{x}}$ contains the time derivative of the representation of orientation instead of the angular velocity. The geometric Jacobian matrix $\boldsymbol{J}_{\mathrm{g}}(\boldsymbol{q})$ can be derived by a geometric formula or by performing the partial derivative $\boldsymbol{J}_{\mathrm{g}}=\partial\boldsymbol{v}(\boldsymbol{q},\dot{\boldsymbol{q}})/\partial\dot{\boldsymbol{q}}$ based on the kinematics of velocities of the rigid bodies in the robot. The latter is beneficial for hybrid robots.

Test 2. The implementation of the Jacobian can be tested as follows: Let $\boldsymbol{q}_1$ and $\Delta\boldsymbol{q}$ be arbitrary vectors in $\mathbb{R}^n$ with $\|\Delta\boldsymbol{q}\| \ll 1$ and $\boldsymbol{q}_2=\boldsymbol{q}_1+\Delta\boldsymbol{q}$. The pose difference $\Delta\boldsymbol{x}=\boldsymbol{x}_2-\boldsymbol{x}_1=\boldsymbol{f}(\boldsymbol{q}_2)-\boldsymbol{f}(\boldsymbol{q}_1)$ can also be obtained by using the Jacobian with $\Delta\boldsymbol{x}'=\boldsymbol{J}(\boldsymbol{q}_1)\Delta\boldsymbol{q}$. The dash only denotes the second implementation. If the implementation of the Jacobian is correct, we have $\|\Delta\boldsymbol{x}-\Delta\boldsymbol{x}'\|<\varepsilon$. This relation is trivial from a mathematical point of view, since the test only uses differential calculus from the derivation of $\boldsymbol{J}$ in property 2. However, this has to be explicitely implemented as a test to ensure the correctness of the implementation of $\boldsymbol{J}(\boldsymbol{q})$. Throughout this paper the threshold $\varepsilon \approx 10^{-9}$ is used to check for numeric equality. This accounts for linearization error within differential relations and rounding errors of floating point numbers in the numerous operations.

Property 3. The *inverse dynamics* equation $\boldsymbol{\tau}=\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}}+\boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}})+\boldsymbol{g}(\boldsymbol{q})$ for the rigid body robot model in joint coordinates of the robot gives torques $\boldsymbol{\tau}$ of the joints required to perform the motion $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, $\ddot{\boldsymbol{q}}$. The general equation describes both prismatic and revolute joints. The inertia matrix $\boldsymbol{M}$ takes inertial couplings into account, $\boldsymbol{c}$ denotes the vector of centrifugal and Coriolis forces and

$\boldsymbol{g}$ contains the influence of gravitational effects. The equation can be obtained either by the Newton-Euler algorithm or the Lagrangian equations of the second kind.

Test 3. Both methods are implemented in the proposed toolbox. Doing this allows to compare the results of the two implementation by $\|\boldsymbol{\tau}-\boldsymbol{\tau}'\|<\varepsilon$. For the numeric test, random numbers are used for joint positions $\boldsymbol{q}$, velocities $\dot{\boldsymbol{q}}$ and accelerations $\ddot{\boldsymbol{q}}$ as well as for kinematic parameters (lengths, constant angles) and dynamics parameters (masses, center of masses, inertia).

Test 4. Several *properties of the dynamics equations* can be exploited to perform further tests on the implementations of the single terms. A Coriolis matrix $\boldsymbol{C}$ can be obtained from the mass matrix $\boldsymbol{M}$ using Christoffel symbols. It follows the relation $\boldsymbol{c}'(\boldsymbol{q},\dot{\boldsymbol{q}})=\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$. This can be exploited by the test $\|\boldsymbol{c}-\boldsymbol{c}'\|<\varepsilon$.

Test 5. The term $\dot{\boldsymbol{M}}(\boldsymbol{q},\dot{\boldsymbol{q}})-2\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$ has to be a skew matrix.

Test 6. The *kinetic energy* can be derived symbolically using mechanics principles as $E_{\mathrm{kin}}(\boldsymbol{q},\dot{\boldsymbol{q}})$. The mass matrix (in generalized coordinates $\boldsymbol{q}$) is connected with the kinetic energy via $E'_{\mathrm{kin}}=\dot{\boldsymbol{q}}^{\mathrm{T}}\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}}$. Comparing the two implementations leads to the test inequality $\|E_{\mathrm{kin}}-E'_{\mathrm{kin}}\|<\varepsilon$.

Test 7. The gravitational model can be tested with a *forward dynamics* simulation. An ODE simulation is performed for $\ddot{\boldsymbol{q}}=-\boldsymbol{M}^{-1}(\boldsymbol{q})[\boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}})+\boldsymbol{g}(\boldsymbol{q})]$ using the Runge-Kutta numerical integration `ode45`. This gives a time series $\boldsymbol{q}(t)$, $\dot{\boldsymbol{q}}(t)$ and $\ddot{\boldsymbol{q}}(t)$. The *sum of energies* over this trajectory is calculated using the potential energy $E_{\mathrm{pot}}(\boldsymbol{q})$, which has to be implemented within the Lagrange approach. The test now checks, if the sum of energies $E_{\mathrm{total}}=E_{\mathrm{kin}}+E_{\mathrm{pot}}$ stays constant, by using the inequality $\|E_{\mathrm{total}}(t=0)-E_{\mathrm{total}}(t=t_{\mathrm{end}})\|<\varepsilon$.

Property 4. The dynamics equations can be formulated in different *sets of parameters*: barycentric parameters $\boldsymbol{p}_{\mathrm{B}}$ (mass, center of mass, inertia), inertial parameters $\boldsymbol{p}_{\mathrm{I}}$ (mass, first and second moments of mass) and a minimal parameter vector $\boldsymbol{p}_{\mathrm{M}}$ which is a linear combination of the inertial parameters, regrouping parameters with the same effect on the dynamics. The latter implementation is very efficient and essential for the identification of the parameters of a real robot. The former approaches are more intuitive. The inertial parameters only occur in a linear relation in the dynamics equations, allowing to write $\boldsymbol{\tau}'=\boldsymbol{\Phi}_{\mathrm{I}}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})\boldsymbol{p}_{\mathrm{I}}$ and $\boldsymbol{\tau}''=\boldsymbol{\Phi}_{\mathrm{M}}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})\boldsymbol{p}_{\mathrm{M}}$. This *identification model of the dynamics* requires a specific approach to the derivation of velocity and energy.

Test 8. To compare the different implementations regarding sets of parameters, a consistent set of parameters $\boldsymbol{p}_{\mathrm{B}}$, $\boldsymbol{p}_{\mathrm{I}}$ and $\boldsymbol{p}_{\mathrm{M}}$ is created using the parallel axis theorem. Then the tests $\|\boldsymbol{\tau}-\boldsymbol{\tau}'\|<\varepsilon$ and $\|\boldsymbol{\tau}-\boldsymbol{\tau}''\|<\varepsilon$ are performed.

Test 9. It is tested numerically if $\boldsymbol{\Phi}_{\mathrm{M}}\boldsymbol{p}_{\mathrm{M}}$ is a minimal form of $\boldsymbol{\Phi}_{\mathrm{I}}\boldsymbol{p}_{\mathrm{I}}$. Via QR decomposition it is checked that the information matrix for a virtual identification problem with random virtual trajectory samples $\boldsymbol{q}_1,\dot{\boldsymbol{q}}_1,\ddot{\boldsymbol{q}}_1,\boldsymbol{q}_2,...$ has $\mathrm{rank}([\boldsymbol{\Phi}_{\mathrm{I}}^{\mathrm{T}}(\boldsymbol{q}_1,\dot{\boldsymbol{q}}_1,\ddot{\boldsymbol{q}}_1),\boldsymbol{\Phi}_{\mathrm{I}}^{\mathrm{T}}(\boldsymbol{q}_2,\dot{\boldsymbol{q}}_2,\ddot{\boldsymbol{q}}_2),...]^{\mathrm{T}})=\dim(\boldsymbol{p}_{\mathrm{M}})$.

Property 5. The *internal cut forces* from the rigid body dynamics are calculated with $\boldsymbol{w}^{\mathrm{T}}=[\boldsymbol{w}_0^{\mathrm{T}}, \boldsymbol{w}_1^{\mathrm{T}}, ..., \boldsymbol{w}_n^{\mathrm{T}}]^{\mathrm{T}}$, where $\boldsymbol{w}_i$ contains the stacked cut force and cut moment for rigid body i (cut at the corresponding joint) from the Newton-Euler approach.

Property 6. The geometric approach for the $6\times n$ Jacobian matrix $\boldsymbol{J}_{\mathrm{g}}$ can be extended to obtain a $6\times 6(n+1)$ cut force Jacobian $\boldsymbol{J}_{\mathrm{g,cut}}$. The internal cut forces from an external wrench (force and moment) can be obtained using this matrix with $\boldsymbol{w}'=\boldsymbol{J}_{\mathrm{g,cut}}^{\mathrm{T}}(\boldsymbol{q})[\boldsymbol{f}_{\mathrm{ext}}^{\mathrm{T}}, \boldsymbol{m}_{\mathrm{ext}}^{\mathrm{T}}]^{\mathrm{T}}$.

Test 10. Further, it has to be ensured that the implementations of kinematics and dynamics match each other. For the test, we set $\ddot{\boldsymbol{q}}=\dot{\boldsymbol{q}}=\boldsymbol{0}$ and set only one mass of the robot to be non-zero. Here, $\boldsymbol{f}_{\mathrm{ext}}=\boldsymbol{f}_{\mathrm{grav}}$ is the force resulting from the test mass gravity and $\boldsymbol{m}_{\mathrm{ext}}=\boldsymbol{0}$. Both expressions for the cut force have to be identical, which can be tested numerically with $\|\boldsymbol{w}-\boldsymbol{w}'\|<\varepsilon$.

3.2 Serial-Hybrid Robots

Serial-hybrid robots consist of a serial main structure connecting the base and the end effector with additional closed kinematic loops, as depicted in Fig. 1,b. The closed loops are used to constrain degrees of freedom or to shift the position of motors within the structure [21]. The kinematics of closed loops require a different approach than of open loops. The joint coordinates are separated into the generalized (active joint) coordinates $\boldsymbol{q}$ and passive joints coordinates $\boldsymbol{\theta}$. The theory can be viewed in detail in the textbooks [24,12] and e.g. in [21,13,26,9,6].

Property 7. The default approach uses *loop equations in the implicit formulation* $\boldsymbol{h}(\boldsymbol{q},\boldsymbol{\theta})=\boldsymbol{0}$. For simple mechanisms, such as the planar parallelograms in Fig. 1,b, an *inverse geometric model* can be formulated explicitely as $\boldsymbol{\theta}=\boldsymbol{\theta}(\boldsymbol{q})$. The kinematic model of serial-hybrid robots is set up with the extended version of the modified DH parameters [12,3] taking the branching in the kinematic tree structure into account. Additionally to the open loop model, the loop closing conditions are modeled as symbolic equations for $\boldsymbol{h}(\boldsymbol{q},\boldsymbol{\theta})$ and $\boldsymbol{\theta}(\boldsymbol{q})$ by hand.

Test 11. While creating the model, visual plausibility is checked with a kinematic sketch as in Fig. 1,b.

Test 12. After this, the test $\|\boldsymbol{h}(\boldsymbol{q},\boldsymbol{\theta})\|<\varepsilon$ is performed. The passive joint coordinates $\boldsymbol{\theta}(\boldsymbol{q})$ are obtained symbolically or – if not possible – numerically using the Newton-Raphson algorithm on $\boldsymbol{h}=\boldsymbol{0}$. The kinematic parameters and test configurations for $\boldsymbol{q}$ can not be chosen randomly as in the serial robot case, but have to be chosen as plausible values by visual inspection or from CAD data.

Property 8. The *kinematic constraints* can be formulated in the *differential form* $\dot{\boldsymbol{\theta}}=\boldsymbol{J}_{\boldsymbol{\theta}}\dot{\boldsymbol{q}}$. The constraints Jacobian $\boldsymbol{J}_{\boldsymbol{\theta}}$ can be obtained from the implicit form $\boldsymbol{h}$ of the constraints as $\boldsymbol{J}_{\boldsymbol{\theta}}=-(\partial\boldsymbol{h}/\partial\boldsymbol{\theta})^{-1}(\partial\boldsymbol{h}/\partial\boldsymbol{q})$. Using the elimination approach [25], $\boldsymbol{\theta}(\boldsymbol{q})$ is available in symbolic form and the differential relation $\boldsymbol{J}_{\boldsymbol{\theta}}'=\partial\boldsymbol{\theta}/\partial\boldsymbol{q}$ [21] can be obtained.

Test 13. The two implementations (implicit and explicit form) are tested with the identity of $\boldsymbol{J}_{\boldsymbol{\theta}}$ and $\boldsymbol{J}'_{\boldsymbol{\theta}}$ up to rounding errors ε.

Property 9. Creating the robot model requires the definition of an open-loop tree structure with the coordinates $\boldsymbol{q}_{\mathrm{OL}}$, which contains the coordinates $\boldsymbol{q}$ and $\boldsymbol{\theta}$. For this model, all tests and definitions from Sec. 3.1 can be used.

Test 14. Velocities of the rigid bodies of the robot are now generated by both models based on property 2. For the elimination approach $\boldsymbol{v}=\boldsymbol{J}_{\mathrm{g}}(\boldsymbol{q})\dot{\boldsymbol{q}}$ and for the open-loop structure (implicit approach) $\boldsymbol{v}'=\boldsymbol{J}_{\mathrm{g,OL}}(\boldsymbol{q}_{\mathrm{OL}})\dot{\boldsymbol{q}}_{\mathrm{OL}}$ is used. The entities $\boldsymbol{q}$ and $\boldsymbol{q}_{\mathrm{OL}}$ as well as $\dot{\boldsymbol{q}}$ and $\dot{\boldsymbol{q}}_{\mathrm{OL}}$ are chosen consistently with random numbers like in Test 12. Using this within the test $\|\boldsymbol{v}-\boldsymbol{v}'\|<\varepsilon$ proves the validity of the implementation of the velocities within the algorithm. The velocity and Jacobians can be set up for any rigid body of the mechanism, not limited to the end effector link.

The same approach can be performed for the accelerations.

Property 10. The *dynamics equations* are again deduced by two different approaches to allow testing the results. Using the elimination approach [25], the passive joints $\boldsymbol{\theta}$ are completely eliminated from the symbolic equations already at the kinematics stage. The Lagrangian equations of the second kind are used to deduce the dynamics $\boldsymbol{\tau}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})$ in the closed-loop robots minimal coordinates $\boldsymbol{q}$. The projection approach leads to $\boldsymbol{\tau}'=\boldsymbol{\tau}_{\boldsymbol{q}}+\boldsymbol{J}_{\boldsymbol{\theta}}\boldsymbol{\tau}_{\boldsymbol{\theta}}$, where $\boldsymbol{\tau}_{\boldsymbol{q}}$ and $\boldsymbol{\tau}_{\boldsymbol{\theta}}$ are the components of the open-loop dynamics $\boldsymbol{\tau}_{\mathrm{OL}}(\boldsymbol{q}_{\mathrm{OL}},\dot{\boldsymbol{q}}_{\mathrm{OL}},\ddot{\boldsymbol{q}}_{\mathrm{OL}})$ corresponding to the entries of $\boldsymbol{q}$ and $\boldsymbol{\theta}$ in $\boldsymbol{q}_{\mathrm{OL}}$.

Test 15. The implementations are again tested numerically via $\|\boldsymbol{\tau}-\boldsymbol{\tau}'\|<\varepsilon$. Random values for $\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}$ and consistent values for $\boldsymbol{\theta},\dot{\boldsymbol{\theta}},\ddot{\boldsymbol{\theta}}$ are selected.

Some other tests on the dynamics from Sec. 3.2, such as the test of energy consistency, are also applied.

3.3 Parallel Robots

Parallel robots, as given in Fig. 1,c have a similar modeling approach as hybrid robots since they also contain closed kinematic loops. A detailed overview on the dynamics is given in [3]. Usually the platform coordinates $\boldsymbol{x}$ are chosen as minimal coordinates of the system [19]. The relation between platform velocity $\boldsymbol{v}$ and the time derivative $\dot{\boldsymbol{x}}$ of the platform coordinates has to be regarded in the algorithm [19,1] and is considered in the implementation, but is omitted here for the sake of brevity and only entities related to $\dot{\boldsymbol{x}}$ are presented.

Property 11. For the symbolic derivation of the dynamics the two-step projection approach from [1] with a claim on high efficiency is used and gives the dynamics $\boldsymbol{\tau}_{\boldsymbol{x}}(\boldsymbol{x},\dot{\boldsymbol{x}},\ddot{\boldsymbol{x}},\boldsymbol{q}_{\mathrm{OL}})$ and the inverse Jacobian matrix $\boldsymbol{J}^{-1}(\boldsymbol{x},\boldsymbol{q}_{\mathrm{OL}})$ with $\dot{\boldsymbol{q}}=\boldsymbol{J}^{-1}\dot{\boldsymbol{x}}$. The dynamics $\boldsymbol{\tau}_{\boldsymbol{x}}$ in platform coordinates can be projected into the active joint coordinates with $\boldsymbol{\tau}=\boldsymbol{J}^{\mathrm{T}}\boldsymbol{\tau}_{\boldsymbol{x}}$. This represents the actuator force necessary to achieve the robot motion given by $\boldsymbol{x}$, $\dot{\boldsymbol{x}}$ and $\ddot{\boldsymbol{x}}$ – a value necessary for simulation and control.

Property 12. This implementation is verified by a second approach, which is taken from [5] and presents a more general approach than standard algorithms [19]. It mainly corresponds to a general form of the state-of-the-art approach for the dynamics of closed kinematic loops from [6] with focus on using the platform coordinates $\boldsymbol{x}$. The *implicit definition of the constraints* $\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{q}_{\mathrm{OL}})=\boldsymbol{0}$ is only partially implemented symbolically due to the high computational demand in the general case. Again, $\boldsymbol{q}_{\mathrm{OL}}$ includes the active joint coordinates $\boldsymbol{q}$ and the passive joints coordinates $\boldsymbol{\theta}$ of the kinematic leg structure (without the platform).

Property 13. The *differential formulation* $\boldsymbol{h}_{\boldsymbol{x}}\dot{\boldsymbol{x}}+\boldsymbol{h}_{\boldsymbol{q}_{\mathrm{OL}}}\dot{\boldsymbol{q}}_{\mathrm{OL}}=\boldsymbol{0}$ with $\boldsymbol{h}_{\boldsymbol{x}}=\partial\boldsymbol{h}/\partial\boldsymbol{x}$ and $\boldsymbol{h}_{\boldsymbol{q}_{\mathrm{OL}}}=\partial\boldsymbol{h}/\partial\boldsymbol{q}_{\mathrm{OL}}$ can be obtained from the constraints equations. This leads to the *inverse Jacobian matrix* $\boldsymbol{J}_{\mathrm{OL}}^{-1}=-\boldsymbol{h}_{\boldsymbol{q}_{\mathrm{OL}}}^{-1}\boldsymbol{h}_{\boldsymbol{x}}$ for the full joint vector, relating $\dot{\boldsymbol{q}}_{\mathrm{OL}}=\boldsymbol{J}_{\mathrm{OL}}^{-1}\dot{\boldsymbol{x}}$. Selecting only the rows corresponding to the active joint coordinates gives the inverse Jacobian matrix $\boldsymbol{J}'^{-1}$, where the dash only marks the second implementation in demarcation of the first one.

Test 16. The gradient matrices $\boldsymbol{h}_{\boldsymbol{x}}$ and $\boldsymbol{h}_{\boldsymbol{q}_{\mathrm{OL}}}$ are tested against the constraints formulation $\boldsymbol{h}$ by defining $\Delta\boldsymbol{x}$ and $\Delta\boldsymbol{q}_{\mathrm{OL}}$ with $\|\Delta\boldsymbol{x}\|\ll 1$ and $\|\Delta\boldsymbol{q}_{\mathrm{OL}}\|\ll 1$. Let $\boldsymbol{x}_1$ and $\boldsymbol{q}_{\mathrm{OL},1}$ be arbitrary random numbers with $\boldsymbol{h}_1=\boldsymbol{h}(\boldsymbol{x}_1, \boldsymbol{q}_{\mathrm{OL},1})\neq\boldsymbol{0}$. The values $\boldsymbol{x}_2=\boldsymbol{x}_1+\Delta\boldsymbol{x}$, $\boldsymbol{q}_{\mathrm{OL},2}=\boldsymbol{q}_{\mathrm{OL},1}+\Delta\boldsymbol{q}_{\mathrm{OL}}$ and $\boldsymbol{h}_2=\boldsymbol{h}(\boldsymbol{x}_2, \boldsymbol{q}_{\mathrm{OL},2})$ are calculated. As a second step, $\boldsymbol{h}'_2=\boldsymbol{h}_1+\boldsymbol{h}_{\boldsymbol{x}}\Delta\boldsymbol{x}+\boldsymbol{h}_{\boldsymbol{q}_{\mathrm{OL}}}\Delta\boldsymbol{q}_{\mathrm{OL}}$ is calculated and the two implementations are tested with $\|\boldsymbol{h}_2-\boldsymbol{h}'_2\|<\varepsilon$. This of course is (again) mathematically trivial, but necessary, as elaborated upon in test 2. Due to the complexity of the terms the implementation is otherwise prone to errors.

Test 17. Both implementations $\boldsymbol{J}^{-1}$ and $\boldsymbol{J}'^{-1}$ from properties 11 and 13 are tested for equality up to rounding errors of ε within the numerical computation.

Similar tests can be defined for the second time derivative of the the constraints equation, which is used to determine the acceleration relations.

Property 14. The second implementation of the *dynamics of parallel robots* [5] is determined numerically. The approach is very similar to the case of hybrid robots using the constraints Jacobians [24,6]. Expressed in platform frame, it results $\boldsymbol{\tau}'_{\boldsymbol{x}}=\boldsymbol{\tau}_{\mathrm{P}}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \ddot{\boldsymbol{x}})+\boldsymbol{J}_{\mathrm{OL}}^{-\mathrm{T}}(\boldsymbol{q}_{\mathrm{OL}}, \boldsymbol{x})\boldsymbol{\tau}_{\mathrm{OL}}(\boldsymbol{q}_{\mathrm{OL}}, \dot{\boldsymbol{q}}_{\mathrm{OL}}, \ddot{\boldsymbol{q}}_{\mathrm{OL}})$. The dependencies on $\boldsymbol{q}_{\mathrm{OL}}$ and $\boldsymbol{x}$ and their time derivatives are added for clarity. These quantities have to fulfill the constraints equations $\boldsymbol{h}=\boldsymbol{0}$, $\dot{\boldsymbol{h}}=\boldsymbol{0}$ and $\ddot{\boldsymbol{h}}=\boldsymbol{0}$. The dynamics of the platform as a rigid body in Cartesian space is considered with the term $\boldsymbol{\tau}_{\mathrm{P}}$ and $\boldsymbol{\tau}_{\mathrm{OL}}$ contains the open-loop dynamics of the single leg chains that are deduced with the methods presented in Sec. 3.1.

Test 18. Both implementations are then tested using the inequality $\|\boldsymbol{\tau}_{\boldsymbol{x}}-\boldsymbol{\tau}'_{\boldsymbol{x}}\|<\varepsilon$.

Other tests for the dynamics, such as energy consistency by time-integration of the forward dynamics, can be performed as presented in Sec. 3.1.

4 Description of the Proposed Toolchain

The fundamentals of kinematics and dynamics of Sec. 3 are implemented in a toolchain to obtain the robot models for serial, hybrid and parallel robots following the requirements introduced at the end of Sec. 1. The program is structured within several Maple worksheets which each contain only a limited set of fundamental equations, corresponding to one of the numbered properties in Sec. 3. This allows a convenient debugging with the graphical user interface of Maple. All intermediate symbolic expressions are exchanged between worksheets via data files which are saved in one worksheet and read in the next. Therefore each worksheet can run independently, once previous parts are generated. This structural decision can be justified also by the experiences with another toolbox [5], which was implemented solely based on Maple procedures. This made debugging and extending the tool an impossible task regarding 280 interleaved procedures.

The proposed toolchain has three workflows corresponding to the robot type, where the *serial robot* case is the most central one. The workflow for *parallel robots* is modular. It first generates the corresponding serial kinematic leg chain which is then used by the the approach of prop. 11 in Sec. 3.3. The use of the Lagrange equations of the second kind allows a modular reuse of the worksheets for serial robots also for *hybrid robots* using the elimination approach. The second implementation for hybrid robots of prop. 10 in Sec. 3.2 is implemented in a modular way similar to parallel robots using the workflow of the open-loop tree structure first and then applying the worksheets for the implicit constraints.

The overall workflow is summarized in Fig. 3. As *step 1*, serial robots are described with an input definition file using DH parameters from [12] and parallel robots by an additional definition file referring to the leg chain and alignment of the base joint. For hybrid robots, a separate manually created worksheet for the constraints of Sec. 3.2 is necessary. *Step 2* comprises (automatically) running all Maple worksheets. To enable batch and partially parallel processing, all worksheets (`.mw` files) are saved separately in a text format (with `.mpl` extension), which can be run by the terminal application of Maple. Every worksheet exports the symbolic expressions of the model equations as optimized code in Matlab syntax using the Maple `CodeGeneration` package. Following basic principles for software quality [32], in *step 3* the automatically generated optimized Matlab code for all symbolic expressions is post-processed to reach a certain standard. A Bash script creates a function file with a header comment with short description of the function and its inputs and outputs, assertions to prevent unexpected user input, compiler information, statistics of the code generation and finally the optimized code itself. After all function files are (automatically) generated, the unit test framework is run in *step 4*. If all tests are passed, the results can be used for their designated purpose in *step 5*.

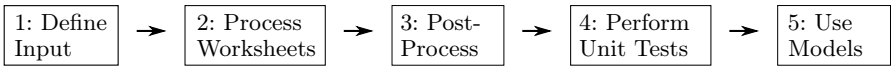


Fig. 3. Overview of the overall workflow of the toolchain

5 Application to a Model Database Framework

As introduced in Sec. 1, one of the purposes of the toolchain is to create an algorithm that is able to determine the best robot for a given task. In a first step of structural synthesis all possible structures have to be identified and their description stored systematically. Then all robot models have to be generated in symbolic form. Finally, the batch-optimization of all robots is performed and the best robot is selected. All serial kinematic chains are generated using the approach from [23], which is similar to the evolutionary morphology from [11]. Leg chains for parallel robots are created with the same approach with modifications on the requirements from parallel robot structural synthesis [15,11]. This leads to a database of the kinematic descriptions, i.e. Denavit-Hartenberg parameters for serial robots [12]. For parallel robots lists of possible leg chains and alignments of the base and platform coupling joints are stored. The databases are saved in text-based `csv`-tables to facilitate software version control using Git.

The models in the serial robot database are created as Matlab functions by batch-processing the robot definition files with the proposed toolchain. This stored input and output data, referenced in the footnotes on p. 3 of the paper, can be regarded as a case study for the validity of the toolchain. At the current stage, 616 unique kinematic chains (with 3 to 6 joints) are stored, which by elimination of isomorphisms represent all possible structures [23]. The database contains approximately 36 thousand Matlab files with 6 million lines of automatically generated code. The size of the complete database is around 300 MB and therefore still feasible. Generating all robot models takes about 5 days of CPU time on a standard desktop computer. Due to partially parallel execution, all model files for one serial robot can be generated within one hour. By also generating the whole database with the tools SymPyBotics [27] and OpenSymoro [14], the inverse dynamics of property 3 is validated against another reference. The number of operations of these different implementations are counted in the generated code and are compared in Fig. 4. The proposed Maple toolchain (with Newton-Euler) has a similar efficiency as the Python-based references. Using Lagrange is less efficient, as expected from literature [12].

Code generation for parallel robots shows that the selected symbolic approach [1] of property 11 in Sec. 3.3 is very efficient for simple kinematic structures with a

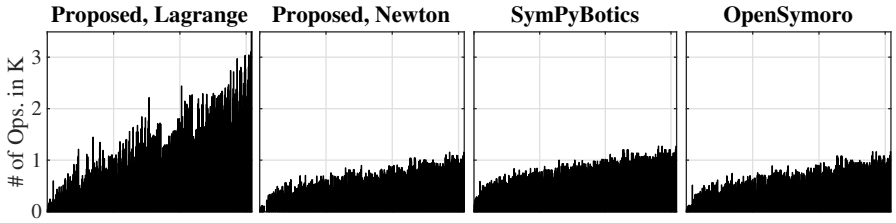
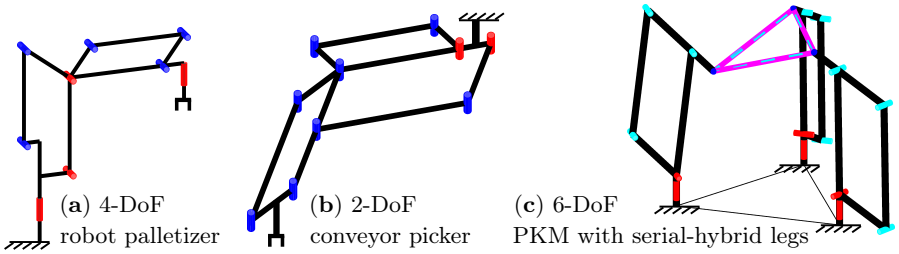


Fig. 4. Comparison of the number of operations for inverse dynamics of the proposed toolbox with two methods (Lagrange, Newton-Euler) and two reference toolboxes (SymPyBotics, OpenSymoro) over all 616 serial kinematics on the horizontal axis. Sorted by increasing total number of joints and then by number of revolute joints.



(**a**) 4-DoF robot palletizer (**b**) 2-DoF conveyor picker (**c**) 6-DoF PKM with serial-hybrid legs

Fig. 5. Examples of hybrid robots implemented in the program framework.

low number of kinematic parameters (resulting from a low number of mechanical joints). For general kinematic leg chains with revolute joints instead of universal and spherical joints, the necessary computation time can reach several days and is not feasible any more. The case of parallel robots with platform coupling joints that are not spherical is not included in the symbolic approach [1]. In summary, only symbolic code for 91 parallel robot models was included in the database, which in total consists of a few thousand symmetric parallel robots with 3 to 6 platform degrees of freedom (DoF). To perform dynamics simulations for parallel robots therefore the approach [5] of property 12 in an extended formulation is used, allowing also robots with non-spherical coupling joints.

Only some examples of industrial robots with closed loops are implemented manually, such as robot palletizers (see Fig. 1,b or Fig. 5,a) and the 2-DoF pick-and-place machine for conveyor belts from Fig. 5,b. An automated systematic synthesis of serial-hybrid robots or parallel robots based on serial-hybrid leg chains as (the manually created example) in Fig. 5,c is not implemented yet.

6 Conclusion

The presented new toolchain for robot dynamics stands out against existing tools by focusing on an integral approach of a complete workflow from a robot definition to a stand-alone dynamics model implementation. Quality requirements, such as automatic documentation and testing, are explicitly considered. No additional steps have to be performed by the user, such as manually post-processing toolbox output or testing the results beyond integration tests. This allows the deployment as a model generator for an extensive robot database which is used for a dimensional synthesis over all existing robots to find the best robot for a specified task.

Acknowledgements

This work was developed over five years using funding from the German Research Foundation (DFG, grant 341489206), the Federal Ministry of Education and Research of Germany (BMBF, grant 16SV6175) and the European Union's Horizon 2020 research and innovation programme (grant 688857).

References

1. Abdellatif, H., Heimann, B.: Computational efficient inverse dynamics of 6-DOF fully parallel manipulators by using the Lagrangian formalism. Mechanism and Machine Theory **44**(1), 192–207 (2009). https://doi.org/10.1016/j.mechmachtheory.2008.02.003
2. Bethge, S., Malzahn, J., Tsagarakis, N., Caldwell, D.: FloBaRoID – a software package for the identification of robot dynamics parameters. In: Int. Conf. on Robotics in Alpe-Adria Danube Region. pp. 156–165. Springer (2017). https://doi.org/10.1007/978-3-319-61276-8_18, project homepage: https://github.com/kjyv/FloBaRoID
3. Briot, S., Khalil, W.: Dynamics of Parallel Robots, Mechan. Machine Science, vol. 35. Springer (2015). https://doi.org/10.1007/978-3-319-19788-3
4. Corke, P.: Robotics, vision and control. Springer Tracts in Advanced Robotics (2011). https://doi.org/10.1007/978-3-642-20144-8, the CodeGenerator extension was added by Jörn Malzahn, available at https://github.com/petercorke/robotics-toolbox-matlab/tree/master/@CodeGenerator; a Python version of the toolbox is under development at https://github.com/petercorke/robotics-toolbox-python
5. Do Thanh, T., Kotlarski, J., Heimann, B., Ortmaier, T.: On the inverse dynamics problem of general parallel robots. In: IEEE Int. Conf. on Mechatronics. pp. 1–6 (2009). https://doi.org/10.1109/ICMECH.2009.4957202
6. Docquier, N., Poncelet, A., Fisette, P.: Robotran: a powerful symbolic gnerator of multibody models. Mechanical Sciences **4**(1), 199–219 (2013). https://doi.org/10.5194/ms-4-199-2013, https://www.robotran.be/
7. Featherstone, R.: Rigid Body Dynamics Algorithms. Springer Science & Business Media (2008). https://doi.org/10.1007/978-1-4899-7560-7
8. Felis, M.L.: RBDL: an efficient rigid-body dynamics library using recursive algorithms. Autonomous Robots pp. 1–17 (2016). https://doi.org/10.1007/s10514-016-9574-0, project homepage: https://rbdl.github.io
9. Fisette, P., Postiau, T., Sass, L., Samin, J.C.: Fully symbolic generation of complex multibody models. Mechanics of Structures and Machines **30**(1), 31–82 (2002). https://doi.org/10.1081/SME-120001477
10. Gogu, G.: Families of 6R orthogonal robotic manipulators with only isolated and pseudo-isolated singularities. Mechanism and Machine Theory **37**(11), 1347–1375 (Nov 2002). https://doi.org/10.1016/S0094-114X(02)00048-4
11. Gogu, G.: Structural Synthesis of Parallel Robots, Part 1: Methodology, vol. 866. Springer (2008). https://doi.org/10.1007/978-1-4020-5710-6
12. Khalil, W., Dombre, E.: Modeling, Identification and Control of Robots. Hermes Penton Science (2002). https://doi.org/10.1016/B978-1-903996-66-9.X5000-3
13. Khalil, W., Creusot, D.: Symoro+: A system for the symbolic modelling of robots. Robotica **15**(2), 153–161 (1997). https://doi.org/10.1017/S0263574797000180
14. Khalil, W., Vijayalingam, A., Khomutenko, B., Mukhanov, I., Lemoine, P., Ecorchard, G.: OpenSYMORO: An open-source software package for symbolic modelling of robots. In: IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics. pp. 1206–1211. Besançon, France (Jul 2014). https://doi.org/10.1109/AIM.2014.6878246, https://github.com/symoro/symoro
15. Kong, X., Gosselin, C.M.: Type synthesis of parallel mechanisms. Springer Publishing Company, Incorporated (2007). https://doi.org/10.1007/978-3-540-71990-8
16. Krefft, M.: Aufgabenangepasste Optimierung von Parallelstrukturen für Maschinen in der Produktionstechnik. Ph.D. thesis, Technische Universität Braunschweig, Germany (2006), ISBN 3802786890, Vulkan-Verlag GmbH

17. Kuo, C.H., Dai, J.S.: Structural synthesis of serial robotic manipulators subject to specific motion constraints. In: Proc. of the ASME 2010 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Montreal (2010). https://doi.org/10.1115/DETC2010-28947
18. Kurz, T., Eberhard, P., Henninger, C., Schiehlen, W.: From Neweul to Neweul-M^2: Symbolical equations of motion for multibody system analysis and synthesis. Multibody System Dynamics **24**(1), 25–41 (2010). https://doi.org/10.1007/s11044-010-9187-x, project homepage: https://www.itm.uni-stuttgart.de/software/neweul-m/
19. Merlet, J.P.: Parallel robots, Solid Mechanics and Its Applications, vol. 128. Springer S. & B. Media, 2nd edn. (2006). https://doi.org/10.1007/1-4020-4133-0
20. MSC Software Corporation: MSC Adams, website, accessed Dec. 2nd 2020, https://www.mscsoftware.com/product/adams
21. Nakamura, Y., Ghodoussi, M.: Dynamics computation of closed-link robot mechanisms with nonredundant and redundant actuators. IEEE Transactions on Robotics and Automation (1989). https://doi.org/10.1109/70.34765
22. Nethery, J.F., Spong, M.W.: Robotica: a Mathematica package for robot analysis. IEEE Robot. & Automat. Mag. **1**(1), 13–20 (1994). https://doi.org/10.1109/100.296449, github.com/RoboticSwarmControl/robotica
23. Ramirez, D.A.: Automatic generation of task-specific serial mechanisms using combined structural and dimensional synthesis. Ph.D. thesis, Gottfried Wilhelm Leibniz Universität Hannover, Germany (2018). https://doi.org/10.15488/4571
24. Samin, J.C., Fisette, P.: Symbolic modeling of multibody systems, Solid Mechanics and Its Applications, vol. 112. Springer Science & Business Media (2003). https://doi.org/10.1007/978-94-017-0287-4
25. Schappler, M., Lilge, T., Haddadin, S.: Kinematics and dynamics model via explicit direct and trigonometric elimination of kinematic constraints. In: Proc. of the 15th IFToMM World Congress (2019). https://doi.org/10.1007/978-3-030-20131-9_311
26. Shi, P., McPhee, J.: Dynamics of flexible multibody systems using virtual work and linear graph theory. Multibody System Dynamics **4**(4), 355–381 (2000). https://doi.org/10.1023/A:1009841017268
27. Sousa, C.D., Cortesão, R.: SageRobotics: open source framework for symbolic computation of robot models. In: Proc. 27th Annu. ACM Symp. Applied Computing. pp. 262–267 (2012). https://doi.org/10.1145/2245276.2245329, project homepage (successor): https://github.com/cdsousa/SymPyBotics
28. Tedrake, R., the Drake Development Team: Drake: Model-based design and verification for robotics (2019), project homepage: https://drake.mit.edu
29. The MathWorks, Inc.: Matlab Simscape Multibody, website, accessed Dec. 2nd 2020, https://mathworks.com/products/simmechanics.html
30. Wang, Y., Gondokaryono, R., Munawar, A., Fischer, G.S.: A convex optimization-based dynamic model identification package for the da Vinci research kit. IEEE Robotics and Automation Letters **4**, 3657–3664 (2019). https://doi.org/10.1109/LRA.2019.2927947, project homepage: https://github.com/WPI-AIM/dvrk_dynamics_identification
31. Waterloo Maple Inc.: MapleSim, website, accessed Dec. 2nd 2020, https://maplesoft.com/products/maplesim/
32. Wilson, G., Aruliah, D.A., Brown, C.T., Hong, N.P.C., Davis, M., Guy, R.T., Haddock, S.H., Huff, K.D., Mitchell, I.M., Plumbley, M.D., et al.: Best practices for scientific computing. PLoS biology **12**(1), e1001745 (2014). https://doi.org/10.1371/journal.pbio.1001745